Universidad
Carlos III de Madrid

Diseño e implementación de un automóvil autónomo a escala comunicado con control remoto

Trabajo Final de Grado

Ingeniería Informática

Autor:

Alberto Carrera Martín

Tutor:

Javier Fernández Muñoz

Madrid, Febrero del 2017

Agradecimientos

Como agradecimientos, en primer lugar me gustaría mencionar a toda mi familia, en especial a mis padres y a mi hermano, por haber estado siempre ahí, ayudándome en todo lo que pudiesen y por su infinita paciencia.

También querría hacer una mención especial a Laura, sin ti hubiera sido mucho más costoso anímicamente la tarea de hacer un trabajo tan duradero y que requiere tanto esfuerzo.

A mi tutor, ya que sin él la multitud de ideas y soluciones a los problemas que fueron surgiendo no hubiese sido tan efectiva.

Tampoco me puedo olvidar de toda mi gente de Carabanchel y de la universidad, sin vosotros no hubiese tenido tantos buenos momentos que me ayudaron a quitar el estrés acumulado.

Muchas gracias.

Abstract

This document has several objectives, which all of them need to be completed to ensure the correct fulfillment of the work which takes place here. One of this aims is the collection of all the information that has played a role in the realization of the entire project from simple guides or templates of concrete aspects to thorough investigations on more difficult topics that emerged during the development. Another of its objectives is the written development of the fundamental sections for the accomplishment of this work, which will be explained next, such as the analysis of the requirements, the implementation of the system, etc.

It includes, moreover, for a better understanding, a wide table of contents and two indexes that comprises the illustrations that are present in the document, as well as the tables that have been created during the project.

Introduction

The document, firstly, has an introduction about the subject that occupies this work. It briefly and generally presents the topic of radio frequency and autonomous vehicles; it deals with their use and their conception within the real world.

Secondly, the principal objective of the work is explained. The aim of the work has been based on the use of an open-source hardware, focusing specifically on Arduino. From here, the base is a platform destined to the creation of a personalized hardware with a world of possibilities. Analogously, taking a look at the engine world, it is perfectly possible to create an electronic system linked to this field which is, at the same time, related to Arduino. This led to the analysis of different possibilities.

There were already some projects that dealt with line-following vehicles, others that worked with sounds, etc., but something that would not seem to exist was a vehicle that followed a transmitter through the use of radio frequency which, in a completely autonomous way, was able to triangulate the position of the transmitter with the objective of going towards it. Due to the unrealistic aspect of this case, it was chosen as the starting point.

Analyzing the topic, it was proved viable, since there are communication devices by radio frequency at a low cost and it belongs to the technological investigation field, allowing in this way the discovery of alternative ways of doing already implemented things like GPS, a technology that allows to position objects with a system based on satellites and triangulation, in a very effective way.

Unlike regarding the GPS, this project would provide a 'homemade' triangulation by radio frequency, which would not need, for instance, the extremely high costs that the GPS setting entailed. The only necessary things would be a transmitter and several receivers which can communicate with each other.

Objectives

The principal objective that constitutes this study is to show the evidence that, with the simple use of communication by radio frequency, it is possible to triangulate the transmitters of that signal. The most commonly used method for this is the Global Positioning System 'GPS'; however, the case of this study presents an alternative which is, moreover, low cost.

Another principal aim is the task of construction the structure of a vehicle which allows the resulting electronic system to move and to have support.

Secondary objectives are the ones that have taken place in the development of the project which are divided in six chapters. Among them we find: introduction, state of the art, analysis and design of the system, implementation and tests of the system, planning and budget, and conclusion and future research. At the beginning of this summary, the introduction was already defined, so in the following part the rest of the chapters will be explained, pointing out their contents and the purposes.

State of the Art

In this second chapter, it takes place a detailed analysis of the state of the technology used in this project, being these the open-source platforms, and mainly Arduino. On the one hand, the chapter contains an analysis of the open-source platforms that are the most important and the most frequently used globally nowadays, with the aim of showing which of them is the best with respect to the objective of this project.

Finally, from them, Arduino was selected, since it provided everything that was necessary for the making of this project: a wide variety of components with different functions, several boards which are distinctive of Arduino, and a development environment with its own language. The chosen board was Arduino Mega 2560, due to the fact that it is the one which has a bigger quantity of connections to components, which would allow, if necessary, to increase the amount of elements that are connected to each of the boards.

Analogously, an analysis that deals with hardware components that were compatible with Arduino and whose purchase would be possible was carried out. The principal objective was to provide the vehicle with a structure. Because of this reason, a research on specific Arduino chassis was made. In addition, it was necessary to purchase radio frequency components, since they are the basis which the project was based on. On the other hand and not less important, it was necessary to find a controller for the engines, apart from a sensor of obstacles in order to avoid collision.

Finally, this chapter refers to an additional software that was used during the project's development, and it also includes a study of the alternative software and hardware that were present during the analysis of the viable elements for this work.

Analysis and Design of the System

This third chapter includes a complete analysis of all the requirements present in the system, which will help in the final design and implementation. In this way, it can be shown that all the requirements are accomplished, which implies that the prototype's objectives have been correctly carried out.

Firstly, it is necessary to obtain the user requirements, which symbolize the conditions that have to be achieved in order for the system to be valid. From them, software requirements are obtained; they present a technical description of the requirement in question. Once all the requirements are obtained, the traceability matrix must be completed in order to confirm, firstly, that the corresponding software requirements have been obtained from all the user requirements, and, secondly, to confirm which are the relations between them.

Likewise, there is a study about the different cases of use that can be present in the system for the purpose of showing which are these cases and which steps must be taken in order to solve the scenes in question.

Once the analysis of the whole system is finished, the design is composed. This design has to satisfy all the requirements for it to be valid, and in that way, its implementation can be carried out. The design is divided into hardware and software:

Hardware design

Each of the hardware elements used is described in terms of functioning. Besides, there is a description of the resulting connection scheme, depending on the type of pin that each component has and on the place of the board where it is connected.

The first described element is the chassis of the vehicle, on which it offers a graphical and written explanation about how the engines should be connected.

After, there is the controller of engines, whose main characteristic is the possession of two internal circuits called " Bridge H ", which help to alternate the current provoked on the engines to make the turns of the wheels possible towards both sides, depending on if the vehicle has to go towards ahead, to turn, etc.

Later, it is exposed the design of the kits of communication by radio frequency. The most important thing in this section is the specification of how there will work the communication that will take place among the issuer and all the receptors, which is the base of all this project. In this way, the guidelines of behavior depending on the current scene between the components and the purpose of the action that continues in each of them are presented.

Ending with the principal components, there is exposed the ultrasonic sensor, which is the responsible for the vehicle to not being struck with the obstacles along all his way. For this component, there is an explanation on which it is detailed how this curious component works internally and in what way it turns into information the effect of the waves that produces on the outside.

To complete this section, it is explained how it will be the design of the present feedback from the two systems, receiver and transmitter. Though in the practice it is a simple scene of several LEDs issuing light depending on some conditions, it is

something of a crucial importance inside the computer science, since it is essential that the user knows as far as possible what happens within the system.

Software design

Once the functioning of each hardware component and their connection scheme are clear, the controller that will guide them can be established in the way that the system expects it to do it. For each end of the connection (in one end, the transmitter, and in the other the vehicle), there is a controller responsible for establishing their behavior, according to the requirements indicated in the analysis of the system. These controllers are designed in this section and created in the phase of implementation and tests of the system, which takes place after this section.

Implementation and tests of the system

In this paragraph it is attached the documentation of the whole phase of implementation with the final tests of the system to verify his correct running.

The implementation is divided in two sections, on one hand the implementation of the controller of the issuer and, on the other hand, the controller of the vehicle. The implementation of the issuer was the simplest in the developing process, due to the fact that only it is necessary to control the issuing component of radio frequency in order to send messages constantly.

The controller of the vehicle was what involved practically all the time of this section of the project. In this controller it was necessary the implementation of several tasks, being these the following ones:

Anti-crash system

Thanks to the measurements given by the component, there is known the distance to which the next objects are on the direction of the vehicle. With this information, in case the obstacle is too close, the anti-crash system will be activated, avoiding the object.

Messages reception system

By means of the frequency receptor, an obtaining of messages is carried out individually for each one. As they cannot work simultaneously, the active receptor is alternated between them, until each receptor has tried to obtain an exact number of messages. Finally, there is a number of messages that each one has received, which will serve to calculate the direction. As an extra of safety, a specific string was set in the message, which also knows the issuer in order to verify that the one who is transmitting

messages is an authentic issuer. Only the identity of the issuer might be usurped if the chain was also known by other, or has obtained the string by other ways.

Calculation of the direction system

Due to the information calculated in the previous step, there are 4 values that represent the messages that came to every receptor on the maximum messages that they tried to read. Through these values, we obtain the percentage of messages that each one had, so the value of every receptor is divided by the total number of messages that were read among the four receptors. Later, these percentages are saved in a two-dimensional array, due to the fact that it is necessary the second row to support the reference with the receptor from which the percentage comes, and then it is ordered from highest to lowest depending on the percentage.

Once organized, in an assumed beginning, the receptor would be chosen by major percentage, unless if the percentages are similar, which would imply that the vehicle is next to the receptor, so it should stop and wait for it to take distance. On this calculation of the direction, a fail-safe system was also implemented, because, unfortunately, the receptors do not have the insured communication, and it is possible that a receptor, in a concrete moment, doesn´t detect any message though it is near to the issuer. In the worst case, it might happen that the receptor placed further of the issuer, is the one that has read more messages, and the calculation would give a wrong goal direction.

For these cases, there is had an array of two positions to guard the last two taken directions. In case the direction taken in the current iteration is totally opposite to the stored previous one, it will take as a not valid direction and will return to take the stored direction. In the opposite case, if the new direction agrees with the previous one, it will take like correct. If the generated direction doesn't agree with the previous one, it doesn't take, but if in the following iteration is the same one, it takes for valid because of the repetition.

Activation of the engines system

Finally, there is the implementation that takes charge of, depending on the generated and verified direction, moving the engines towards the mentioned direction, or stopping the vehicle.

About this global implementation, later the due unitary tests were carried out to verify that every implemented function has a correct behavior of agreement to the aims marked for each of them.

Every checking tests the cases that can make fail the system, beside covering all the requirements that were obtained in the phase of analysis. To demonstrate the fulfillment of each one of the requirements, a traceability matrix appears between the tests and the own functional requirements.

Planning and Budget

First of all, this chapter deals with the temporal plan that each of the tasks will take. For a better understanding, a Gantt diagram is added. This diagram shows an itemization of each task, indication how much time was employed until their finalization.

Secondly, the budget necessary to carry out the project is presented and it is divided according to its origin. The budget is divided into staff costs, in terms of hours and salary, software costs, and finally hardware costs. Regarding hardware costs, a calculation was made to find out the amortization of temporary products, in order to make the expenses more precise. At the end of this chapter, it is showed the complete budget about all the project.

Conclusion and Future Research

This final chapter covers, on the one hand, general and concrete conclusions derived from the project, mentioning the accomplished objectives and results. On the other hand, the chapter comprises a personal opinion about the preparation of this project as well as of any general project of this significance.

Finally, the chapter gives details of improvements that might be introduced on the already-made system for the purpose of providing new functionalities or improving the existing ones in order to create a most complete system.

Results

The results derived from this work were finally positive. The vehicle responded correctly to the environment; thus, the principal function of the receiver, the tracking task, was working according to expectations. When moving away from the vehicle, the

movement was correct in most of the cases; when situated at a minimum distance, the vehicle remained still since it had reached the receiver. This was achieved after a long process of implementation of the triangulation by radio frequency, carried out by the simple connection by radio frequency between a transmitter and four receivers located equably in space.

Moreover, in this connection, it was also possible to differentiate messages from the authentic transmitter by means of the delivery from the transmitter and the acknowledgment from the receivers of a concrete string of characters only known by each of the parts.

Furthermore, the vehicle also responded correctly regarding the collision system. When it came across an obstacle, the anti-collision system started to function, avoiding obstruction and, finally, allowing the vehicle to continue its tracking task.

Another satisfying result was the fact that, since it was an uninsured connection (even though transmitter and receiver are at a suitable distance, the message can be not received, due to the fact that the active receiver is constantly changed without waiting for the message to be received), the possibility of the worst case was supported. This situation was that, after tracking a specific direction, suddenly the calculation of the next direction led to the opposite of the current one. This is possible because the receiver located at the furthest point from the transmitter, due to a scarce existing probability, can be the one that receives more messages from the connection.

If this was the case, the system would ignore the calculated direction, but it would store it. If, after calculating the next direction, the result was the same, the vehicle would take that direction, and, again, if the calculation resulted in the direction previously taken, the vehicle would change directions one more time. Hereby a system has been contributed to test of mistakes where the vehicle will be prevented in the possible major thing from taking bad decisions, something very important in a system that comes to be almost completely autonomous.

Conclusions

The conclusions, as it was explained previously, appear at the end of the document. About them, there are written the general conclusions obtained during the development of the project and the personal conclusions.

If we talk about the general conclusions, it stands out the fact of being able to have overcome the aims marked before starting the idea selected for the project in question. These objectives are all of the study's work, analysis, investigation and implantation of the final system that was obtained, which supposed a long way of development, and problems that caused delays in the completing of every task.

The personal conclusions, on the other hand, are focused at the personal experiences that were reach because of have had to produce this individual work, with the exception of the assistances of the tutor and the manuals about that area that has taken place in this project.

Besides, I can also highlight the personal knowledge acquired when doing such a complex and long project, which helps greatly face the possible and similar future cases.

Keywords: Arduino, Radio Frequency, Triangulation, Vehicle.

Índice

Índice de tablas

Índice de Ilustraciones

Capítulo 1. Introducción

Si se hablase del uso de radiofrecuencia aplicado a un vehículo, vendría a la imaginación una especie de vehículo pequeño que es controlado mediante un mando a distancia, con el cual se indica al auto las acciones que debe realizar, tales como ir hacia delante, girar, dar marcha atrás, etc.

En esta ocasión, la radiofrecuencia que intervendrá en el vehículo servirá para que él mismo compruebe y ejecute los movimientos que considere precisos para lograr el objetivo de llegar hasta el emisor de dicha frecuencia, el cual estará situado en algún punto cercano a los distintos receptores, de manera que exista una comunicación entre ellos. Por tanto, hay un cambio de roles: el vehículo pasa a ser el organismo "pensante" en el escenario.

La complejidad que reside en esta cuestión es el hecho de intentar obtener una aproximación lo más exacta posible de la localización del emisor por medio de la comunicación por radiofrecuencia entre él y los receptores instalados en el vehículo, sin ayuda de tecnologías tales como el GPS que, por ejemplo, se ayuda de satélites repartidos alrededor de todo el planeta con el fin de triangular la posición de una manera extraordinariamente efectiva del objeto solicitante.

En este presente documento, tendrá lugar el proceso de identificación y desarrollo de todo lo necesario para llevar a cabo esta idea, y así demostrar que no se quedaría en una mera hipótesis.

1.1 Objetivos

El principal objetivo de este trabajo es demostrar que, sin necesidad de tecnología especializada en la triangulación de objetos, se puede llevar a cabo dicha triangulación mediante una simple comunicación por radiofrecuencia entre dos elementos, de manera que en este caso se detecte la posición del emisor, y el vehículo vaya por sus propios medios hasta él.

Lo más novedoso de este objetivo es el hecho de crear un sistema completamente autónomo en cuanto a la las acciones que lleva a cabo internamente, excluyendo el hecho de que tiene que haber una parte emisora, el cual le indique por dónde está la dirección sobre la que se tiene que mover.

Para lograr esta tarea, primero se tienen que llevar a cabo otros fines secundarios. Estas metas se detallan a continuación:

1. Diseño y montaje de un vehículo a escala.

Se creará una estructura con los componentes básicos y necesarios para tener un prototipo de lo que sería, una vez completado al 100%, una representación a escala de un vehículo auténtico.

2. Adición de un sistema anti-choques, para conseguir que el vehículo, mientras sigue su ruta, pueda sortear los obstáculos que se presenten.

Habrá que analizar cuál es el componente necesario para esta funcionalidad, y cuál es su forma de implementación en el sistema.

3. Creación de los controladores, tanto para la parte emisora como para la receptora.

Para esto se tendrá que estudiar previamente cómo se desea que actúe el vehículo en función de la comunicación que tenga lugar con el emisor, los componentes extra que se consideren necesarios de incluir y su implementación, todos los requisitos que comprenderán nuestro sistema y los casos de uso, etc. Además, se deberá realizar una comprobación de los posibles fallos que puedan ocurrir y su posterior gestión.

4. Asegurar que el sistema muestre feedback para el usuario.

Un aspecto de los más importantes en la informática es realizar feedback con los procesos internos de un sistema informático hacia el usuario, con el fin de que dicho usuario sepa qué es lo que está pasando en cada momento.

5. Proveer de un sistema a prueba de fallos.

Por lo general, todo sistema informático tiene aspectos más delicados, los cuales pueden ser más sensibles a errores. La tarea en este caso, será analizar qué puede producir errores en el sistema, y cuál es la manera de minimizarlos, añadiendo así gran fiabilidad al sistema.

6. Proteger la comunicación.

En cada una de las comunicaciones que existen en el ámbito de la informática, existen ataques malignos que pueden provocar una gran cantidad distinta de perjuicios para los integrantes de la comunicación. En este caso, al tratarse de una comunicación por radiofrecuencia en el que tiene lugar un paso de mensajes, habrá que estudiar cuáles son las opciones de seguridad, e implementarlas.

7. Planificar cada una de las tareas que tendrán lugar.

Algo muy importante en proyectos de tal envergadura y duración es, el hecho de establecer unas fechas de consecución de cada uno de los cometidos que se han planteado. Parece un objetivo simple, pero al estar sujeto a eventos que no estén contemplados, tales como problemas que surjan, mala predicción de tiempos, etc, suele ser un objetivo difícil de cumplir ampliamente.

1.2 Estructura del documento

En este apartado se expondrán los capítulos existentes en el documento, y una breve descripción de cada uno de ellos.

Capítulo 1. Introducción

En este capítulo se hará una introducción a la temática global del proyecto, además de establecer los objetivos del mismo. Seguidamente se hará un estudio del marco regulador y del entorno socio-económico aplicable a este trabajo.

Capítulo 2. Estado del arte

En esta sección se mostrará un análisis de la tecnología actual referente a este proyecto. Una vez hecho esto y en base a ello, se seleccionará la tecnología usada tanto para la parte de hardware como software, además de presentar las alternativas posibles.

Capítulo 3. Análisis y Diseño del sistema

En este tercer capítulo se van a establecer todos los requisitos y casos de uso que tendrán lugar en el sistema, con el fin de, una vez analizado en su plenitud, proceder más adelante a su diseño e implantación final.

Capítulo 4. Implantación y Pruebas del sistema

En esta parte del documento tiene lugar toda la implantación del sistema. Para asegurar la correcta implantación se llevará a cabo finalmente una fase donde se someterá a prueba todo lo implementado.

Capítulo 5. Planificación y presupuestos

En este capítulo se expondrá toda la planificación de la creación del proyecto, así como todas sus distintas fases. Además, se detallará todo el presupuesto que ha sido necesario para su culminación.

Capítulo 6. Conclusiones y líneas futuras

Finalmente, se enunciarán las conclusiones obtenidas a partir de todo el proceso llevado a cabo en el proyecto, además de un análisis de las posibles mejoras aplicables al proyecto, las cuales añadirían más funciones que ayudarían a completan aún más el producto.

1.3 Marco Regulador

1.3.1 Análisis de la legislación aplicable

Actualmente, las nuevas tecnologías y todos los adelantos de la ciencia que los respaldan, así como la ampliación de conocimientos en los diferentes ámbitos de la informática, ejercen una importante influencia en la vida cotidiana y el entorno.

Todo ello, aplicado al sector laboral, cumple funciones como producir beneficios para cubrir necesidades que, con el avance del conocimiento van surgiendo, asumiendo la legalidad vigente en cuanto a su ejercicio. Además, estas secuencias de acciones que hay que llevar a cabo en cualquier trabajo para conseguir buenos resultados, deben estar justificadas por juicios éticos que partan de una teoría moral, y que a su vez establezca que, efectivamente, lo que se está realizando se está ejecutando de manera adecuada y correcta.

En este sentido, se hace necesario señalar la existencia de deberes y derechos de los que deben responsabilizarse empresarios y trabajadores y que aparecen regulados en el Estatuto de los Trabajadores en su título primero, capítulo uno y segunda sección: derechos y deberes laborales básicos. Por lo tanto, se deduce que ambas figuras tendrán que responder siempre de sus actos en la jornada laboral, y además se esperará de ellos que se centren en su cometido y que estén capacitados para recabar datos, elaborar la información obtenida y proponer alternativas y soluciones al problema surgido.

En definitiva, cualquier trabajador ha de dominar los conocimientos y habilidades necesarias para desempeñar sus tareas bajo una deontología profesional que guíe la

toma de decisiones. Concretamente algunas de las principales responsabilidades éticas que se pueden resaltar en cualquier puesto de trabajo son:

- Elaborar productos de calidad y útiles.
- Respetar el medio ambiente, cuidándolo de cualquier contaminación posible.
- Hacer buen uso de los recursos naturales con los que se puedan contar.
- Realizar el trabajo cumpliendo con la ley y normas vigentes; respetando los contratos laborales.
- Respetar las condiciones de trabajo.
- Favorecer la seguridad y la salud laboral de los trabajadores.

Para que todo esto se cumpla (y haciendo especial hincapié en los dos últimos puntos), surge la prevención de riesgos laborales: consiste en la aplicación de una serie de medidas que velan para descubrir anticipadamente los riesgos que derivan de cualquier puesto de trabajo, y evitar así cualquier posible accidente laboral.

Esta prevención aparece también regulada en el Estatuto de los Trabajadores, en materia de prevención, que en su artículo 4.2 (en la relación de trabajo), recoge el derecho que tienen "a su integridad física y una adecuada política de seguridad e higiene" y en su artículo 19.1 "a una protección eficaz en materia de seguridad e higiene".

Así que, en la legislación actual aparece recogido el derecho de los trabajadores a un trabajo con condiciones de seguridad y salud, que implica a su vez el deber del empresario de dotar a estos trabajadores de las protecciones necesarias para ello. Para conseguirlo, se descompone el puesto de trabajo en las distintas tareas que se llevan a cabo y se tiene en cuenta en qué condiciones se realizan: herramientas empleadas, lugar, tipo de maquinaria usada, etc.

No obstante, cuando se habla de riesgos laborales no sólo se está haciendo referencia a dichos riesgos, sino también a las enfermedades profesionales que surgen en el ejercicio laboral. Estas enfermedades están recogidas en el Real Decreto 1299/2006 de 10 de noviembre; y son todas aquellas producidas como consecuencia de las circunstancias físicas, psicológicas, etc que se derivan del puesto de trabajo.

En el caso de la informática la prevención de riesgos laborales se puede aplicar también a aspectos de seguridad y privacidad relacionadas con el uso de esta tecnología.

Entonces, estableciendo como objetivo al empleo de software (que engloba bases de datos y archivos) y de hardware; se deberán analizar previamente sus problemas, vulnerabilidades y amenazas para poder determinar posibles riesgos derivados de su utilización. Estos riesgos se pueden intentar controlar mediante el uso de la seguridad de la información (una serie de medidas preventivas de los sistemas tecnológicos que permiten proteger la información y la confidencialidad), y la seguridad informática (disciplina que se ocupa de diseñar los procedimientos, normas y técnicas destinadas a conseguir un sistema de información seguro y confiable).

En referencia a los riesgos, se podría decir que, actualmente, prácticamente la mayoría de los coches poseen algún tipo de conectividad como el GPS por ejemplo. Esto implica analizar también la probabilidad de que alguien intercepte esa conectividad del automóvil, y por ende, acceda a datos quizá privados o de otra índole, y nos pueda incluso robar el vehículo.

Por lo tanto, todas estas medidas aplicadas al control remoto de vehículos traen consigo ciertos problemas de seguridad, que pueden permitir que terceras personas ocasionen daños diversos a los vehículos; entre otros, que una persona ajena pueda llegar a averiguar cómo funciona el software de control de ese coche en concreto y encuentre fallos que le permitan hacerse con su control. Por lo que, actualmente, la aplicación del control remoto por radiofrecuencia a los automóviles no estaría preparada para soportar ataques de este tipo.

Sobre los riesgos de seguridad, entre las formas más típicas de ataques al control remoto, se encuentra, por ejemplo, evitar la comunicación entre el vehículo y el propietario, la suplantación, la desactivación de parte del sistema control remoto, transmisión de códigos erróneos, etc.

En cuanto a los riesgos de privacidad de los propietarios de vehículos por control remoto, es importante velar por la información personal, y en este caso las mayores preocupaciones de la población son entre otras el acceso de terceras personas a datos personales o el rastreo de las acciones realizadas; y por tanto la Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal se aplica al uso de radiofrecuencia, así como cada uno de los deberes y derechos que regula, como por ejemplo que se debe llevar a cabo un juicio previo sobre la necesidad de usar este tipo de tecnologías y con qué fines se van a emplear, o que se deberá garantizar la seguridad

de los recursos del software y el hardware relacionados con la tecnología control remoto. Por último, respecto al ámbito de la privacidad cabe destacar la directiva 2009/136/CE7 que establece la necesidad de velar por la protección de los derechos fundamentales, y en particular de la protección de datos, cuando la radiofrecuencia se conecte a redes públicas de comunicaciones electrónicas, y se aplicará también la directiva 2002/58/CE (acerca de la privacidad y la comunicación electrónica) y las disposiciones incluidas sobre seguridad, confidencialidad y datos de tráfico y localización. En este proyecto, al ser únicamente una comunicación en la cual el emisor pulsa un botón transmitiendo una cadena por mensajes desde una posición, no habría riesgo en cuanto a datos personales del usuario.

Para finalizar, es necesario mencionar la regulación del uso de este tipo de tecnología. Así, la Organización Internacional de Normalización (para la creación de estándares internacionales, compuesta por diversas organizaciones nacionales de estandarización) establece algunas normas aplicadas al uso de la radiofrecuencia:

- ISO/IEC 11784-11785, ISO 10536, ISO 18000: sobre privacidad y seguridad a los datos.
- ISO 19762-3: establece los términos y definiciones únicas de identificación por radiofrecuencia en el campo de la identificación automática y captura de datos técnicos.

De todas formas, los gobiernos de cada país miembro de la Organización Internacional de Normalización regularán las frecuencias permitidas, su emisión y su funcionamiento.

1.3.2 Estudio de las cuestiones relacionadas con la propiedad intelectual de la idea

La propiedad intelectual es el conjunto de derechos que corresponden a los autores respecto de sus creaciones, ideas y obras, y es el Ministerio el que se encarga de aplicar la normativa para lograr la protección de la propiedad intelectual. Dentro de la propiedad intelectual se engloban dos tipos de derechos: los del autor y los conexos (que protegen las creaciones de aquellos que, de alguna forma, realizan trabajos y aportaciones a una idea base).

Conociendo los datos anteriores, en este proyecto tiene lugar una idea que podría ser respaldada por los derechos de la propiedad intelectual. En este caso, se habría añadido

un uso más a la radiofrecuencia y control remoto: su aplicación a los automóviles, de manera que de manera autónoma viaje hasta el emisor de dicha señal, algo que hoy en día no está aplicado a nivel global.

Además, se debe tener presente que los controles remotos de radiofrecuencia son utilizados, sobre todo, cuando es necesario un rango de acción superior, donde se necesite además atravesar obstáculos como paredes por ejemplo. De hecho, actualmente ya existen distintas aproximaciones del empleo de la radiofrecuencia en diferentes ámbitos, como por ejemplo los drones, mandos a distancia de puertas automáticas, juguetes de radio control o sistemas de alarma.

Teniendo en cuenta esto, se encuentra a su vez con dos tipos de derechos que quedan englobados bajo la etiqueta "Propiedad intelectual y derechos de autor": los derechos morales y los derechos patrimoniales. De estos, los primeros aparecen más ampliamente recogidos en la legislación española, que incluye aspectos como el reconocimiento de la condición de autor de la obra o el reconocimiento sobre sus ejecuciones (condiciones que no se pueden ceder ni prescriben).

1.4 Entorno Socio-económico

En este siglo actual, y con la creciente sensibilización acerca del respeto hacia la salud y el medio ambiente, se destaca la necesidad de vigilar de cerca las actividades que se realizan, y el desarrollo de las tecnologías para así poder reducir los niveles de contaminación. Por ello, los gobiernos han adoptado diferentes medidas de carácter normativo para minimizar estos efectos negativos, métodos que de hecho han resultado ser de gran eficacia y para ello han precisado analizar su interacción con el medio ambiente y social para así poder aplicar las tecnologías más adecuadas.

El objetivo de la creación de los distintos programas de control de la contaminación ambiental es promover una mejor calidad de vida y por lo tanto requieren la coordinación de las distintas áreas que pueden verse afectadas, como por ejemplo el transporte (que es el problema a analizar que nos importa en este documento y donde podría incidir el uso de un coche por control remoto).

Para poder elaborar estos programas existen diferentes propuestas tales como:

- Establecer una serie de objetivos que se quieren lograr con su aplicación.

- Se establece un recuento aproximado de la cantidad de productos químicos u otros que se van a emitir al medioambiente para poder elegir el programa más adecuado para la prevención de la contaminación.
- Tener en cuenta el uso de los recursos naturales, el consumo de energía y la prevención y el control de la posible contaminación.

Es decir, en general, se sabe que las ondas electromagnéticas interfieren en el desarrollo celular de los seres vivos y pueden causar riesgos o daños a los ecosistemas.

Así que, si se centran estas opciones en el problema que nos ocupa, el uso de automóviles por control remoto constituye una mala alternativa debido, entre otros motivos, a su impacto en la salud pública y ambiental.

Actualmente, muchos productos humanos hacen uso de energía por radiofrecuencia como por ejemplo la radio o el tipo de coche implementado en este trabajo. Estas emisiones de radiofrecuencia pueden ser discutidas en términos de radiación (una propagación de energía en forma de ondas eléctricas y magnéticas), y para valorar su peligrosidad en el ámbito de la salud se debe conocer cómo operan estas energías en nuestro cuerpo: los campos de radiofrecuencia penetran en los tejidos a una profundidad que depende de la frecuencia usada. Esta energía es absorbida por nuestro cuerpo y produce calentamiento.

Diversos estudios han revisado los resultados a corto plazo de la exposición de los cuerpos a la radiofrecuencia y concretamente la OMS ha hecho hincapié en la necesidad de más investigación acerca de la valoración del riesgo de la salud de este tipo de energías: en cuanto a la posibilidad de que aumente el riesgo de cáncer las evidencias empíricas indican que es improbable que la exposición a la radiofrecuencia induzca a la aparición de tumores, a pesar de que por ejemplo un estudio del año 97 encontró que los campos de radiofrecuencia aumentaban la tasa de ratones manipulados genéticamente que desarrollaban leucemia; pero esta implicación en los humanos no está del todo clara. No obstante sí que se han reportado cambios en la actividad cerebral en el tiempo de reacción y los tiempos de sueño, y existe además la posibilidad de que causen interferencias con otros aparatos.

Por el momento, la OMS y otras organizaciones recomiendan:

- Ajustarse a los estándares internacionales de uso de este tipo de energías, mientras se sigue estudiando su impacto en la salud.
- Alejarse de las fuentes electromagnéticas (en función del tipo de aparato la distancia de seguridad varía).
- No permanecer durante mucho tiempo cerca de aparatos que funcionen por radiofrecuencia o fuentes similares y tenerlo por lo tanto en cuenta a la hora de ubicar viviendas, comercios, etc.
- Evitar exponernos a estos campos en las horas de sueño, puesto que se ha encontrado una relación entre ellos y la generación de la melatonina (hormona reguladora del sueño que es sintetizada por el cerebro a la noche).

Capítulo 2. Estado del arte

Hoy en día la idea de la posesión de vehículos reales cuyo funcionamiento pueda ser puramente autónomo es algo únicamente pensado como proyecto a largo plazo. Se lleva investigando mucho tiempo acerca de este ámbito, y en la actualidad básicamente lo que hay son prototipos, e intenciones de llevar a cabo esta idea al mundo cotidiano en un futuro aún lejano, estimado por varios estudios sobre el año 2035.

En el ejemplo de la ilustración 1 se puede ver el caso de un vehículo robotizado, de manera que es capaz de realizar una conducción muy similar a la humana, o incluso mejorarla. Lleva incorporado sistemas tales como GPS, radar o visión computarizada, lo cual supone un gran control del entorno del vehículo.

Ilustración 1. Gummadi, V. (2014). Vehículo robotizado. Recuperado de www.funmonk.com

El principal problema por el que aún no están presentes estos sistemas en nuestras calles es debido sobre todo a que aún la humanidad no está preparada para convivir con ellos. Actualmente, provocaría un caos el hecho de mezclar en las carreteras vehículos que circulan autónomamente con otros que son manejados única y físicamente por personas. Esta hipótesis queda corroborada por los problemas existentes en las conducciones autónomas:

- Dificultad para mantener al conductor despierto.
- Dificultad en la toma de decisiones consecuentes para el conductor.
- Problemas en la ejecución rápida de distintos procesos.
- Comunicación externa.
- El sorteo de obstáculos imprevistos no perfeccionado.

Finalmente, además de este ejemplo de gran proyecto a tamaño real, hay otras tecnologías que permiten simular a escala el comportamiento de un vehículo, sin necesidad de emplear grandes costes. Estas tecnologías son las llamadas plataformas de hardware libre, las cuales se explicarán a continuación.

2.1 Plataformas de hardware libre

En la actualidad, se desconoce mucho acerca de la existencia de este tipo de plataformas en comparación con el software libre. En realidad, el origen de este tipo de proyectos basados en dispositivos de código abierto se remonta a los años 70 en los Estados Unidos, cuando el movimiento "hippie" extendió la famosa frase "Do It Yourself", con las siglas DIY, el cual reivindicaba la fabricación o reparación de cosas por uno mismo, y así de esta manera se ahorraba dinero aprendiendo a la vez. Por lo tanto, rechaza el hecho de tener que comprar materiales que se necesiten para solventar un problema, y en definitiva es una forma de autoproducción basada en nuestras propias ideas y sostenida por la obtención de productos más económicos y personalizados. Fue un movimiento tan sonado que actualmente su ética de trabajo se puede trasladar a cualquier aspecto de la vida cotidiana, incluida el área informática.

Concretamente, el hardware libre terminó de explotar gracias a Internet, lo que ha provocado que a lo largo de los años se haya ido creando y mejorando multitud de diferentes proyectos, con el fin de hacernos más fácil la tarea de crear y personalizar nuestras propias ideas, además de participar en este sector en crecimiento. Ejemplos de estos proyectos son los mencionados a continuación:

- RepRap Project

  Proyecto que tiene sus orígenes en el año 2004, y que tuvo como objetivo la creación de una impresora 3D que fuera capaz de replicarse a sí misma mediante la impresión de casi todos sus componentes. Cualquier persona interesada en este proyecto puede adquirir una impresora 3D de este tipo e imprimir en tres dimensiones un objeto modelado previamente por ordenador, sin necesidad de grandes costes. En la ilustración 2 se puede ver una imagen del proyecto en cuestión.

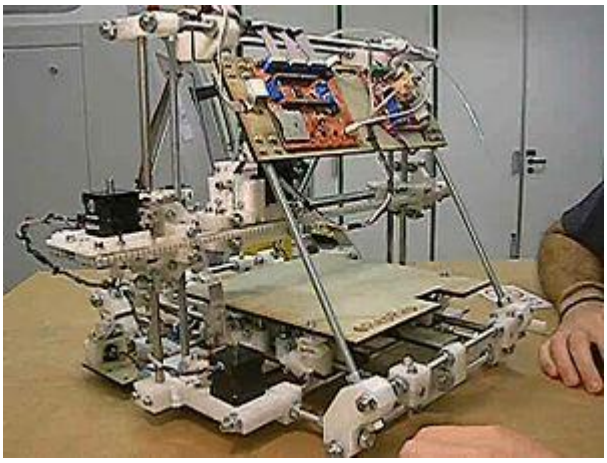

Ilustración 2. (2009). RepRap versión 2.0. Recuperado de www.wikipedia.org

- Proyect Ara

  Proyecto de Google que tiene como objetivo el desarrollo de teléfonos inteligentes modulares. En este caso, la modularidad implica que cada dispositivo móvil está dividido en componentes reemplazables individualmente, tales como la pantalla, el teclado o la batería. Esta ventaja supondría una mayor facilidad para reparar módulos dañados, además de alargar sus ciclos de vida e implicar una reducción en la contaminación provocada por la basura electrónica.

  En la imagen 3 se puede ver un smartphone resultado de este proyecto.

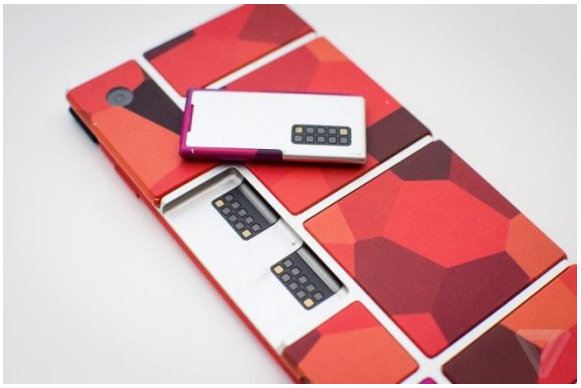

Ilustración 3. (2014). Teléfono inteligente Proyect Ara. Recuperado de www.theverge.com

- Arduino

  Tuvo su origen en el año 2006, creado por un grupo de estudiantes en Italia. Consiste en una placa base, que contiene a su vez un microcontrolador, la cual sumada a su propio entorno de desarrollo de software Arduino IDE da la posibilidad de establecer un controlador para dicha placa, y darle un sentido a su funcionamiento.

Ilustración 4. (2011). Placa Arduino. Recuperado de www.wikipedia.org

Existen multitud de componentes de código abierto que se pueden conectar a la placa, los cuales añadirán funciones determinadas al funcionamiento de la máquina construida. En la ilustración 4 se ofrece una visión acerca de la placa.

- Novena

  Tiene su origen a finales del año 2012, hasta que logró financiación, y por tanto, su desarrollo para su uso libre. Consiste en una placa que se puede adquirir independientemente, integrada en el cuerpo de un portátil, o como ordenador de escritorio, con la cual se obtiene un ordenador en el que se puede controlar absolutamente todo su comportamiento. En la imagen 5 se puede observar un ejemplo de modelo portátil.

Ilustración 5. (2014). Formato portátil de la placa base de Novena. Recuperado de www.hipertextual.com

- Raspberry Pi

  Su origen data del año 2006, donde se empezaron con los primeros diseños de esta plataforma. Finalmente, con la creación de la Fundación Raspberry Pi, se llevó a esta plataforma a multitud de colegios con el fin de dar a conocer y animar al uso de la informática a los estudiantes.

  Consiste en un ordenador de bajo coste, el cual integra memoria RAM, un procesador, y una tarjeta gráfica entre otros componentes, con el que se puede hacer multitud de funciones, tales como ejercer de medidor de energía eléctrica, controlar la temperatura del ambiente, etc. En la imagen 6 se puede ver un ejemplo de la placa en cuestión.

Ilustración 6. (2014). Placa Raspberry Pi. Recuperado de www.protoinformatico.com

2.2 Plataforma de hardware libre seleccionada

En este proyecto, el cual se presentará un vehículo a escala de conducción autónoma, se va a hacer uso de la plataforma Arduino para llevar a cabo tanto la parte de hardware como la de software.

Como se mencionó anteriormente, Arduino es una plataforma de hardware libre, que permite el diseño tanto de hardware como de software y la interacción entre ambos, de manera que puedan cubrir una determinada necesidad. A continuación se desglosa cada una de sus partes, empezando por su hardware y terminando con su software.

2.2.1 Hardware

Por un lado, su hardware tiene como pieza principal una placa característica con un microcontrolador, la cual contiene puertos tanto digitales como analógicos que pueden funcionar como de entrada o de salida de datos. Además, a través de la placa se pueden conectar otros componentes con el fin de conseguir un circuito electrónico con un

objetivo determinado. También dispone de un puerto de conexión USB a través del cual se alimenta la placa, y además se puede establecer la comunicación con el ordenador.

Cada placa se puede alimentar de dos maneras posibles: mediante conexión USB al ordenador, o usando una fuente de alimentación externa (ya sea mediante pilas o baterías). El requisito principal es que el voltaje mínimo sea de 6V como mínimo y 20V como máximo, siendo 12V el recomendado.

Existen diferentes tipos de placas, cada una con diferentes atributos, como por ejemplo la cantidad de pines de conexión, la alimentación necesaria, modelo de microcontrolador, etc. La más básica es la llamada Arduino UNO, pero para este proyecto la placa que se usará será Arduino MEGA 2560 (mostrada en la ilustración 7).

El motivo de selección de esta placa, es principalmente debido a que es la que provee mejor rendimiento, además de ser la que más cantidad de conexiones tiene, y por lo tanto la que más posibilidades ofrece de todo lo que se puede implementar en cualquier sistema Arduino.

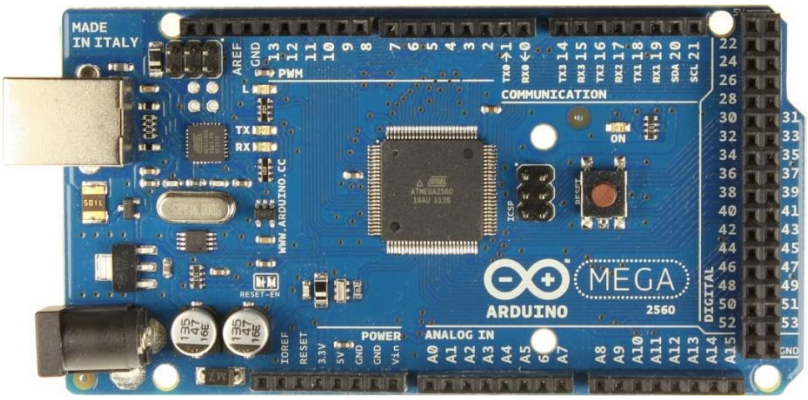

Ilustración 7. Placa Arduino Mega 2560

Centrándonos en sus detalles técnicos, ya se mencionó que dicha placa es considerada de las más competentes de Arduino. En cuanto a pines, tiene en total 54 con posibilidad de uso tanto de entrada como de salida, además de 16 entradas analógicas.

A continuación, se presenta en la tabla 1 con sus características.

Especificaciones	
Microcontrolador	ATmega2560
Voltaje operativo	5V
Voltaje de entrada recomendado	7-12V
Voltaje de entrada límite	6-20V
Pines digitales de entrada y salida	54 (15 con salida PWM)
Pines analógicos	16
Corriente continua por cada pin de E/S	40mA
Corriente continua sobre el pin de 3.3V	50mA
Memoria Flash	256KB (8 usados por el bootloader)
SRAM	8KB
EEPROM	4KB
Velocidad de reloj	16MHz
Largo	101.52mm
Ancho	53.3mm
Peso	37g

Tabla 1. Características placa Arduino Mega 2560

Como se indicó anteriormente, la placa posee dos conexiones para poder alimentarla y hacerle funcionar, como se puede observar en la ilustración 8:

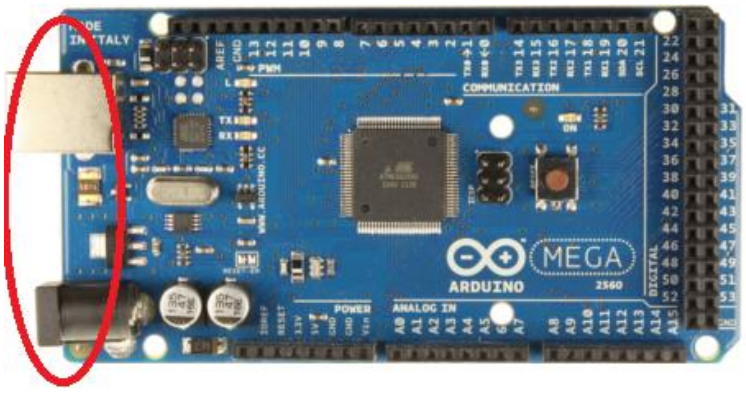

Ilustración 8. Alimentación Arduino Mega

Se tiene el puerto USB ubicado en la esquina superior izquierda, mientras que para emplear una fuente de alimentación externa se deberá usar el conector situado en la esquina inferior izquierda.

Para proporcionar voltaje a partir de la placa, hay varias salidas de corriente detallados en la ilustración 9.

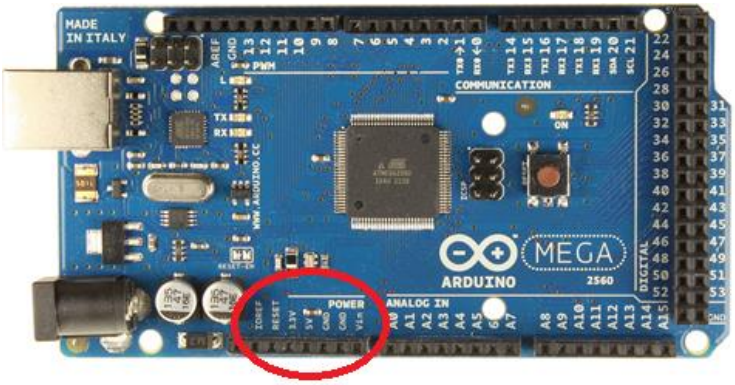

Ilustración 9. Salidas de corriente Arduino Mega

Entre ellos se destacan:

- Vin: se obtiene la misma corriente que entra directamente a la placa mediante su sistema de alimentación.
- GND: toma de tierra
- 5V: se obtiene una tensión de 5V.
- 3.3V: se obtiene una tensión de 3.3V con 50mA como consumo de corriente.
- IOREF: este pin proporciona la tensión con la que opera el microcontrolador.

Como se comentó previamente, la placa dispone de 54 pines de entrada o salida. Su localización se muestra en la ilustración 10.

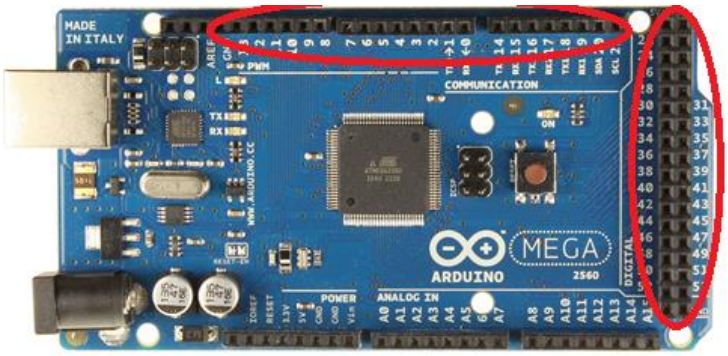

Ilustración 10. Pines E/S Arduino Mega

Además, un alto porcentaje de ellos tienen funciones especiales:

- Función RX: recibir bits en serie. Lo implementan los pines 0, 19, 17 y 15.
- Función TX: enviar bits en serie. Lo implementan los pines 1, 18, 16 y 14.

- Interruptores externos: pines que pueden ser configurados para activar una interrupción a nivel bajo, o tras algún suceso.
- Función PWM: implementada por los pines del 2 al 13, y del 44 al 46.
- SPI: protocolo de datos en serie síncrono utilizado para comunicar un microcontrolador con dos o más dispositivos periféricos rápidamente en distancias cortas, o en su defecto comunicar dos microcontroladores entre sí. Lo implementan los pines del 50 al 53.
- LED: implementada por el pin 13. Consiste en un LED conectado al propio pin, el cual emite luz cuando el pin tiene como valor HIGH, y se mantiene apagado cuando el valor es LOW.
- TWI: implementada por los pines 20 y 21 para soportar la comunicación TWI usando la librería Wire.

Igualmente, la placa también cuenta, como se mencionó antes, de 16 pines de entrada analógica, mostrados en la ilustración 11, cada uno de 10 bits de resolución, lo que da lugar a 1024 diferentes valores de entrada.

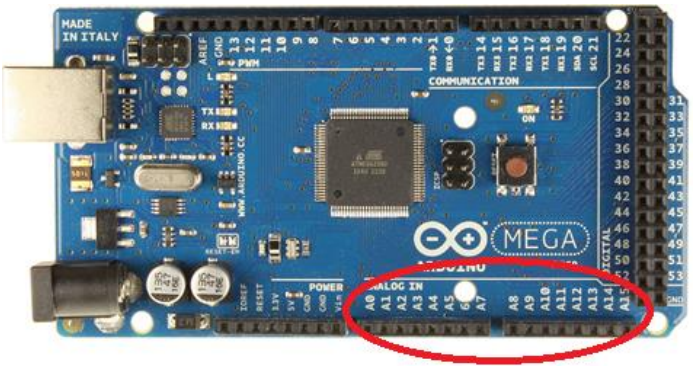

Ilustración 11. Entradas analógicas Arduino Mega

Estos pines sirven básicamente para conectar componentes que introducen valores de entrada en nuestro programa con el fin de tratarlos con el objetivo que tenga lugar.

2.2.2 Software

Por el otro lado, el software lo compone un entorno de desarrollo llamado Arduino IDE, el cual trabaja sobre el lenguaje de programación Arduino.

Arduino IDE es un software completamente gratuito, desarrollado en Java, y dispuesto para que cualquier persona pueda hacer uso de él. Consiste principalmente en un editor de texto, en el cual se crean los programas controladores de las placas

Arduino, para después poder cargarlos en ellas y darles un sentido a su funcionamiento. Además, también posee un menú amplio de opciones, donde se puede establecer, por ejemplo, el puerto USB de la placa donde se quiere cargar el programa, un compilador para comprobar posibles errores en el código, etc.

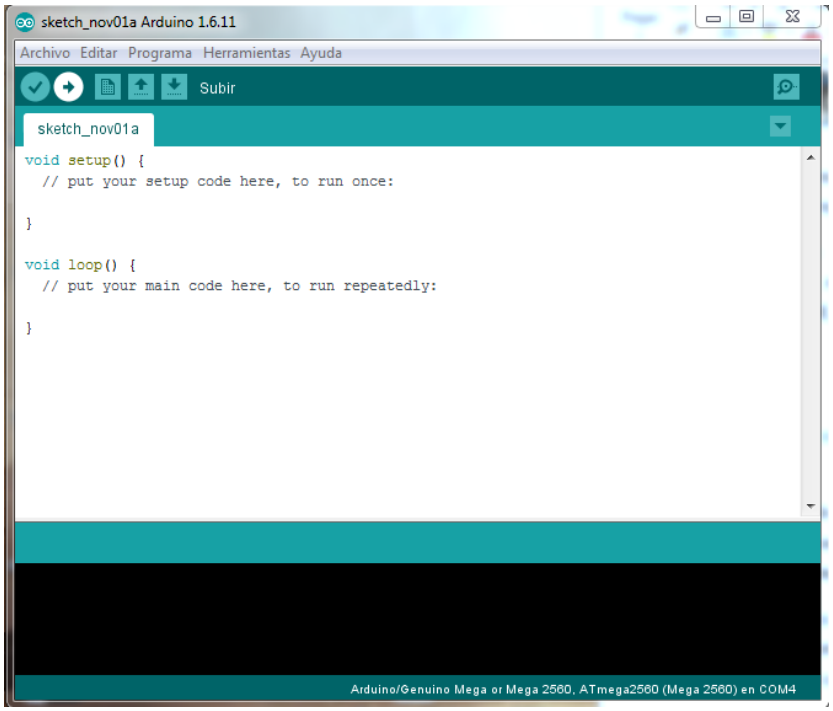

Ilustración 12. Arduino IDE

La imagen 12 muestra un documento en blanco de un sketch cualquiera. Sketch es el término que se da a los programas implementados mediante esta herramienta con el fin de aportar el controlador a la placa Arduino.

Como se puede observar, nada más crear el documento se ven dos métodos distintos. Estos métodos son la base del programa, de uso obligatorio, los cuales se explican a continuación:

- Void setup(): este método únicamente se ejecutará una vez: cada vez que se ejecute el programa en la placa Arduino, y siempre en primer lugar. En él irán todas las instrucciones que sólo necesitan una ejecución en todo el funcionamiento del programa, tales como el establecimiento de pines de entrada/salida por ejemplo. No sería lo óptimo que se ejecutase más de una vez una instrucción que establece, por ejemplo, que el pin 14 de la placa funciona como salida de datos.

- Void loop(): este método, al contrario del anterior, se ejecuta infinitas veces hasta que se detiene el funcionamiento de la placa. A nivel de ejecución, siempre se ejecuta después del setup(), y en él se incluyen todas las instrucciones del programa que es necesario que se repitan, dando así un comportamiento determinado a la placa en cada iteración.

Por su parte, el lenguaje de programación Arduino está basado en el lenguaje de programación C. Como en cada lenguaje, existe un amplio abanico de posibilidades en cuanto a su programación, por lo que, en este apartado, se va a destacar la sintaxis y las estructuras necesarias que servirán para poder confeccionar el programa en su completitud.

- Estructuras de control:
    - Condicionales: if, if-else, switch-case
    - Bucles: for, while, do-while
    - Bifurcaciones y saltos: break, return
- Sintaxis básica
    - Operadores aritméticos: +, -, *, /, %
    - Operadores de comparación: ==. !=, <, >, <=, >=
    - Operador de asignación: =
    - Operadores booleanos: &&, ||, !
    - Delimitadores: ;, {, }
    - Comentarios: //, /*, */
    - Cabeceras: #define, #include
    - Incremento y decremento de variables: ++, --
- Tipos de datos soportados
    - Void, int, unsigned int, char, unsigned char, boolean, string, byte, long, unsigned long, float, double, array
- Constantes:
    - HIGH/LOW: nivel alto y bajo de las señales de entrada/salida
    - INPUT/OUTPUT: establecimiento de pin a entrada o salida de datos
    - True/false: operadores lógicos
- E/S digital
    - pinMode(pin, modo)

    - digitalWrite(pin, valor)
    - digitalRead(pin)
- Tiempo:
    - Delay(ms): instrucción que detiene la ejecución del programa los milisegundos introducidos
- Comunicación por puerto serie:
    - Print()
    - Println()
    - Begin()

2.3 Componentes hardware seleccionados

En este apartado, se expondrá brevemente cada uno de los distintos componentes que han sido seleccionados para confeccionar el prototipo con el fin de cubrir las funcionalidades que se necesita que cumpla. Más adelante, en el capítulo de análisis, diseño e implantación, se detallará más a fondo cada uno de ellos.

2.3.1 Chasis con motores

La base del prototipo, sobre la que irá montado todo el circuito final. Al ser un vehículo a escala, será un chasis pequeño con dos motores sobre el cual se colocará el sistema electrónico una vez montado (como aparece en la ilustración 13).

Para el montaje se tienen los siguientes componentes:

- Base de metacrilato.
- Tornillería.
- 2 ruedas.
- 2 motores.
- Rueda loca.
- Portapilas de 6V.

Ilustración 13. (2015). Piezas del chasis. Recuperado de www.amazon.com

Sus características principales se encuentran en la tabla 2:

Especificaciones	
Dimensiones	22*12*6cm
Voltaje portapilas	6V

Tabla 2. Características del chasis

2.3.2 Motor de corriente continua

Tal como se vio en la descripción de lo que incluía el chasis, este será el tipo de motores utilizados para dotar de movimiento al vehículo. Se trata de un motor de corriente continua, el cual básicamente convierte la energía eléctrica en energía mecánica, produciendo el giro que hace girar a las ruedas.

Lo que diferencia fundamentalmente a este tipo de motores de los de paso a paso, es que éste no es capaz de girar un número específico de grados, algo que aporta una gran precisión a la hora de realizar los movimientos. Un ejemplo de este componente se ofrece en la imagen 14.

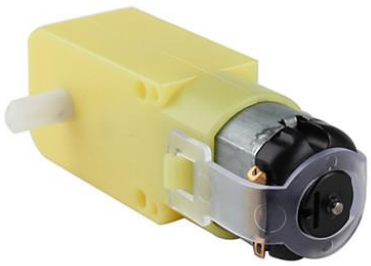

Ilustración 14. (2014). Motor de corriente continua. Recuperado de www.nyplatform.com

Sus características son mostradas en la tabla 3:

Especificaciones	
Voltaje recomendado	Entre 6 y 8V
Voltaje límite	Entre 3 y 12V
Dimensiones	70*22*18mm
Peso	50g

Tabla 3. Características de cada motor

2.3.3 Protoboard

El protoboard es una placa que contiene un circuito interno representado en la ilustración 15:

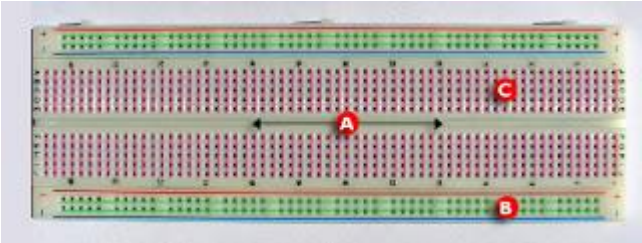

Ilustración 15. (2008). Protoboard. Recuperado de www.circuitoselectronicos.org

Como su propio nombre indica, se trata de una tabla de experimentación, cuya finalidad es la creación de circuitos mediante la conexión de cables y/o componentes electrónicos en él mismo.

Consta de 3 distintas regiones, señalizadas en la propia ilustración:

- Canal central; representado por la letra A. Esta es la región localizada en el medio de la placa, empleada para colocar los circuitos integrados.
- Buses; representados por la letra B. Estos carriles no están conectados entre ellos. Hay dos tipos, los de línea azul, donde se obtiene la toma de tierra, y los de línea roja, donde se obtiene el voltaje que se le dé.
- Pistas; representados por la letra C. Aquí irán conectados los componentes que sean necesarios puentear, para formar el circuito definitivo junto con la placa Arduino.

La tabla 4 muestra sus características principales:

Especificaciones	
Número de buses	50 de voltaje + 50 de toma de tierra
Número de pistas	60 (5 conexiones para cada una)
dimensiones	8*5,5*1cm

Tabla 4. Características del protoboard

2.3.4 Kit RF

Este par de componentes son la pieza clave de este proyecto. Son los encargados de establecer la comunicación por radiofrecuencia entre el emisor (usuario) y receptor (vehículo). Trabajan sobre la frecuencia 433Mhz.

Dicha comunicación es unidireccional, es decir, el módulo emisor siempre será el encargado de enviar la señal, con el fin de que sea recogida por algún módulo receptor. La distancia de transmisión dependerá del voltaje que se le suministre, con valores entre 3 y 12V como límites recomendados.

En la siguiente ilustración (16), se puede ver gráficamente cómo son el dispositivo emisor (derecha), y el dispositivo receptor (izquierda).

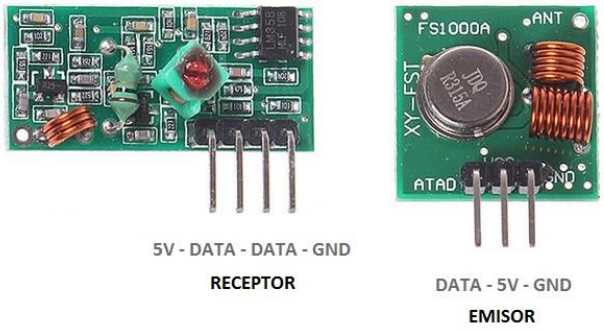

Ilustración 16. Kit RF. Recuperado de www.josehervas.es

Adicionalmente, el uso de antenas amplificaría el alcance de las transmisiones entre ellos.

Las características referentes al módulo emisor aparecen en la tabla 5:

Especificaciones	
Modelo	XY-FST
Voltaje operativo	3V-12V
Consumo	Máximo 40mA y mínimo 9mA
Modo de resonancia	SAW
Modo de modulación	ASK
Frecuencia operacional	433MHz
Alcance	20-200m
Dimensiones	19*19mm
Tasa de transmisión	Menor de 10Kbps
Potencia de transmisión	25mW
Longitud antena recomendada (opcional)	25cm

Tabla 5. Características emisor RF

Siendo, por el otro lado, las siguientes para el módulo receptor (tabla 6):

Especificaciones	
Modelo	RX-RM
Voltaje operativo	DC5V + 0.5V
Consumo	4mA
Método de funcionamiento	OOK/ASK
Dimensiones	30*14*7mm
Frecuencia operacional	433MHz
Tasa de transmisión	Menor de 10KBps
Sensibilidad	100dBm (50Ω)
Longitud antena recomendada (opcional)	25cm

Tabla 6. Características receptor RF

2.3.5 Módulo controlador de motores L298N

Este componente, visualizado en la ilustración 17, se encarga de manejar la movilidad de, o bien dos motores de corriente continua, o bien un único motor paso a paso. Las posibilidades que ofrece en cuanto al movimiento son, por un lado, controlar el sentido del giro de los motores, y por otro la velocidad de giro. Esto lo lleva a cabo gracias a su doble puente H integrado.

Además, lleva su propio regulador de voltaje, así como un controlador de su temperatura.

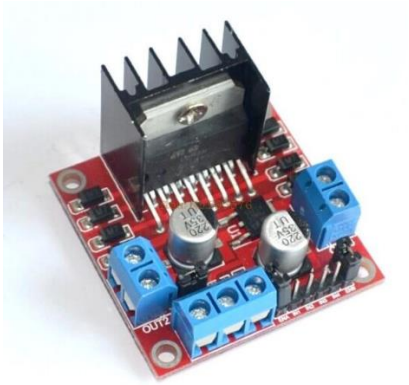

Ilustración 17. Boxall, J. (2014). Controlador de motores L298N. Recuperado de www.tronixlabs.com

Sus características principales se recogen en la tabla 7:

Especificaciones	
Voltaje operativo	5 - 35V
Corriente máxima	2A
Dimensiones	43*23,9*43mm
Regulador de corriente	LM7805
Entradas de control	6

Tabla 7. Características controlador de motores L298N

2.3.6 Sensor ultrasónico HC-SR04

En nuestro proyecto, se necesitará un sensor de distancia que nos indique la existencia de obstáculos en el espacio tal como el de la ilustración 18. Para ello, se usará este componente, el cual es capaz de detectar objetos a cierta distancia mediante la emisión de ultrasonidos que, al rebotar en ellos, regresan al componente dando una pista fundamental acerca de su proximidad.

Ilustración 18. (2014). Sensor Ultrasónico HC-SR04. Recuperado de www.electronicab.com

La tabla 8 muestra sus características:

Especificaciones	
Voltaje operativo	Entre 4.5 y 5.5V
Corriente en reposo	<2mA
Corriente operativa	15mA
Dimensiones	45*20*15mm
Alcance	400cm
Frecuencia de trabajo	40KHz

Tabla 8. Características sensor ultrasónico HC-SR04

2.4 Software adicional empleado

2.4.1 Fritzing para esquemas Arduino

Además del uso del entorno de programación Arduino IDE para el desarrollo del software, se empleará esta herramienta (gratuita también) para diseñar el esquema de conexión de cada elemento hardware con la placa Arduino.

La ilustración 19 representa la ventana por defecto, a partir de la cual se esquematiza la conexión que tenga lugar:

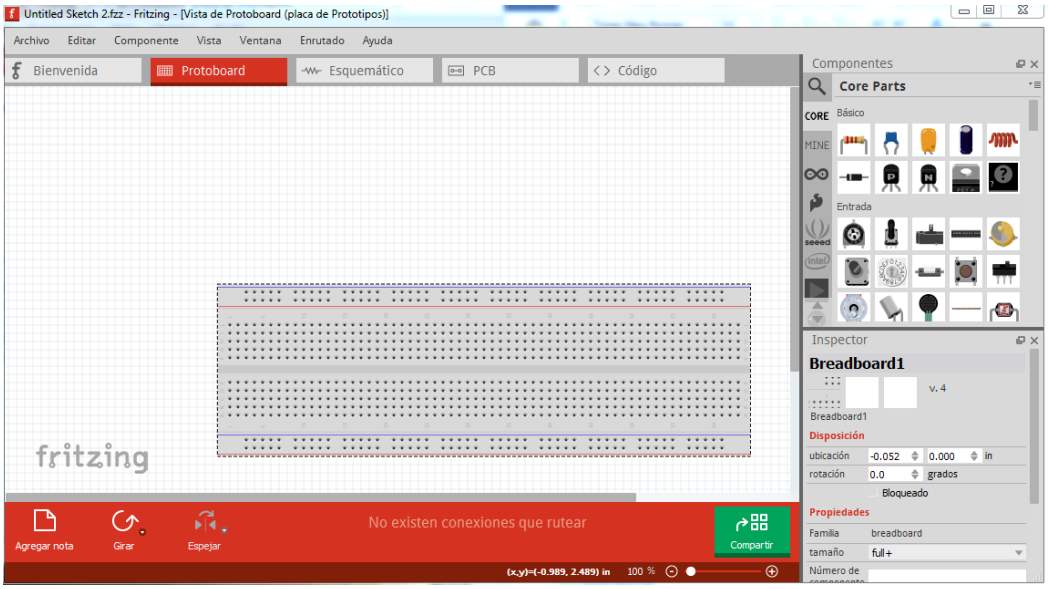

Ilustración 19. Software Fritzing

Como se puede observar, en la ventana mayor aparece el esquema en cuestión, en el cual se irán arrastrando los elementos situados en el menú de la derecha, hasta que, una vez se haya incluido el cableado, se tenga una perfecta ilustración de cómo irá ensamblada esa parte del circuito.

2.5 Alternativas

En este apartado se exponen las principales alternativas encontradas tanto de los componentes hardware seleccionados como del software empleado para el diseño y la implementación del sistema.

2.5.1 Hardware

Sensor de proximidad por ultrasonidos LV-EZ3

Este componente es una alternativa al seleccionado para confeccionar el sistema anti-colisiones. La principal diferencia es que este modelo es capaz de detectar obstáculos hasta una distancia de casi 6.5 metros, además de ser notablemente más caro y aportar más efectividad a la hora de medir las distancias (ilustración 20).

Ilustración 20. (2015). Sensor ultrasónico LV-EZ3. Recuperado de www.bricogeek.com

Comparativa en la tabla 9:

Atributos	LV-EZ3	HC-SR04
Voltaje operativo	Entre 2.5 y 5.5V	Entre 4.5 y 5.5V
Corriente en reposo	<2mA	<2mA
Corriente operativa	15mA	15mA
Alcance	650cm	400cm
Frecuencia de trabajo	42kHz	40kHz

Tabla 9. Comparativa sensores ultrasónicos

Módulo inalámbrico RF APC220

Como alternativa a los módulos de radiofrecuencia seleccionados, se tiene el modelo APC220, el cual presenta una gran mejoría en todos los niveles, ya que cada uno, por ejemplo, puede actuar como receptor o como emisor de señal.

Operan entre las frecuencias 418 y 455Mhz, además de venir con una antena incorporada, la cual incrementa con creces la distancia máxima alcanzable respecto al modelo seleccionado para este proyecto. Como única contra estaría el precio para obtenerlos (ilustración 21).

Ilustración 21. Módulos RF APC220. Recuperado de www.tuelectronica.es

Comparativa en la tabla 10:

Atributos	APC220	Kit RF
Voltaje operativo	3.5-5.5V	3V-12V
Consumo	Máximo 42mA y mínimo 5uA	Máximo 40mA y mínimo 9mA
Modo de modulación	GFSK	ASK
Frecuencia operacional	418-455MHz	433MHz
Alcance	1000m	20-200m
Dimensiones	37.5*18.3*7mm	19*19mm
Potencia de transmisión	20mW	25mW

Tabla 10. Comparativa dispositivos de radiofrecuencia

Módulo controlador de motores Ardumoto

Como su propio nombre indica, este componente se trata de la alternativa al controlador de motores L298N seleccionado para este proyecto.

Este componente, ofrecido en la ilustración 22, está basado en el modelo L298N. De nuevo, presenta la posibilidad de controlar hasta un máximo de dos motores, pero con la diferencia de que puede proporcionar hasta 2 Amperios a cada motor, en lugar de 1, como en el caso del módulo seleccionado. Como funciones adicionales, se destaca la existencia de un LED azul y otro amarillo para indicar el sentido de giro en todo momento.

Ilustración 22. Jimbo. (2015) Placa Ardumoto. Recuperado de www.sparkfun.com

Se muestra la comparativa en la tabla 11:

Atributos	Ardumoto	L298N
Voltaje operativo	4.5-50V	5 - 35V
Corriente máxima	2-3A	2A
Entradas de control	4	6

Tabla 11. Comparativa controladores de motores

Arduino UNO

La placa Arduino UNO es la más básica de toda la gama de placas que posee esta plataforma (mostrada en la ilustración 23). Las diferencias se pueden visualizar en la tabla 12 donde tiene lugar la comparación entre ellas:

Atributos	Arduino UNO	Arduino Mega 2560
Microcontrolador	ATmega328	ATmega2560
Voltaje operativo	5V	5V
Voltaje de entrada recomendado	7-12V	7-12V
Voltaje de entrada límite	6-20V	6-20V
Pines digitales de entrada y salida	14 (6 con salida PWM)	54 (15 con salida PWM)
Pines analógicos	6	16
Corriente continua por cada pin de E/S	40mA	40mA
Corriente continua sobre el pin de 3.3V	50mA	50mA
Memoria Flash	32KB (0.5KB usados por el bootloader)	256KB (8 usados por el bootloader)
SRAM	2KB	8KB
EEPROM	1KB	4KB
Velocidad de reloj	16MHz	16MHz

Tabla 12. Comparativa con Arduino Uno

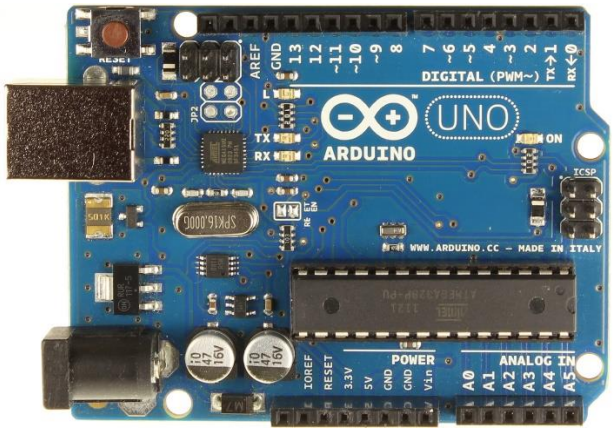

Ilustración 23. Placa Arduino Uno. Recuperado de www.arduino.cc

Arduino Leonardo

Esta placa (cuya imagen aparece en la ilustración 24) está en una situación intermedia respecto a las otras 2. Por un lado, mejora las cualidades de la placa Arduino UNO, pero por el otro lado se ve superada por la placa Arduino Mega. Las diferencias se pueden observar a través de la tabla 13 proporcionada:

Atributos	Arduino Leonardo	Arduino Mega 2560
Microcontrolador	ATmega32u4	ATmega2560
Voltaje operativo	5V	5V
Voltaje de entrada recomendado	7-12V	7-12V
Voltaje de entrada límite	6-20V	6-20V
Pines digitales de entrada y salida	20 (7 con salida PWM)	54 (15 con salida PWM)
Pines analógicos	12	16
Corriente continua por cada pin de E/S	40mA	40mA
Corriente continua sobre el pin de 3.3V	50mA	50mA
Memoria Flash	32KB (4 usados por el bootloader)	256KB (8 usados por el bootloader)
SRAM	2.5KB	8KB
EEPROM	1KB	4KB
Velocidad de reloj	16MHz	16MHz

Tabla 13. Comparativa con Arduino Leonardo

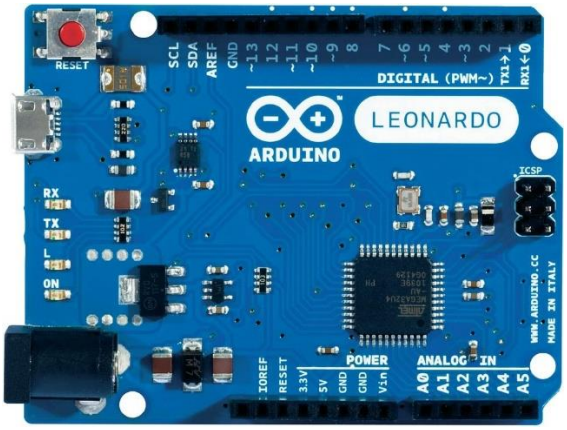

Ilustración 24. Placa Arduino Leonardo. Recuperado de www.arduino.cc

2.5.2 Software

Ardublock

Ardublock es otro entorno de desarrollo para la creación de proyectos Arduino. Cuenta con una herramienta gráfica muy visual de todo lo concerniente al programa que se está implementando. En la ilustración 25 se muestra una imagen donde se puede ver una ligera comparación con Arduino IDE.

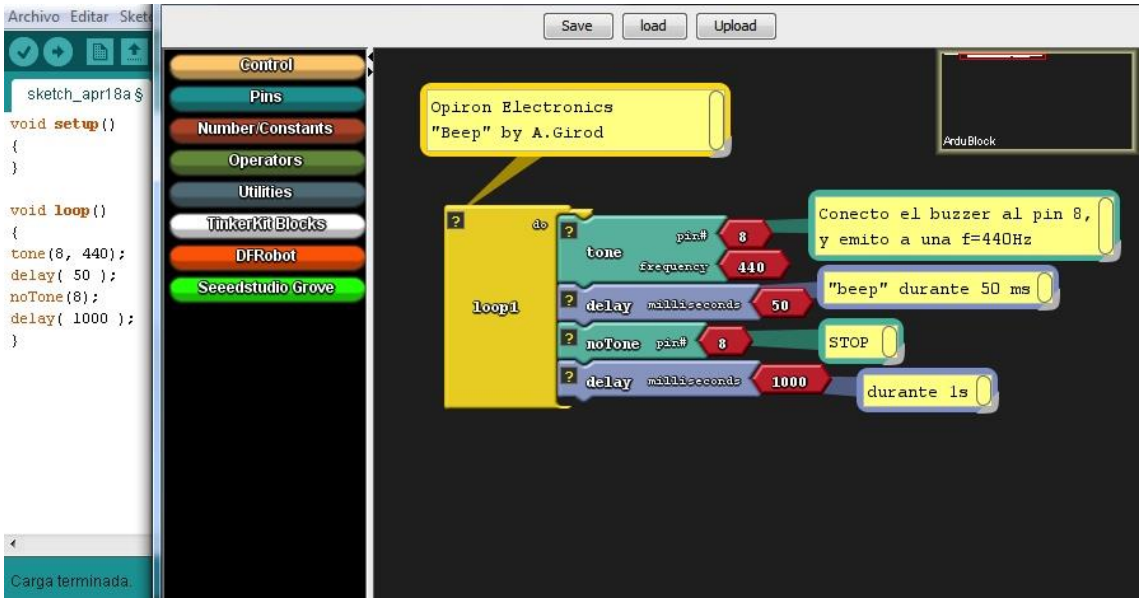

Ilustración 25. Petersen, J. (2016). Comparativa Arduino IDE-Ardublock. Recuperado de www.blog.ardublock.com

Como se puede observar, mientras que con Arduino IDE se establece puramente el código en el programa, mediante Ardublock se puede llegar a lo mismo mediante el uso de bloques con código interno, los cuales una vez que son enlazados forman la estructura del controlador Arduino.

MiniBloq

Por su parte, Minibloq también se aprovecha de la posible ventaja que puede tener trabajar con herramientas gráficas. El uso de este tipo de herramientas u otras como Arduino IDE ya será a gusto del ejecutor. En la ilustración 26 se visualiza cómo es este entorno de desarrollo:

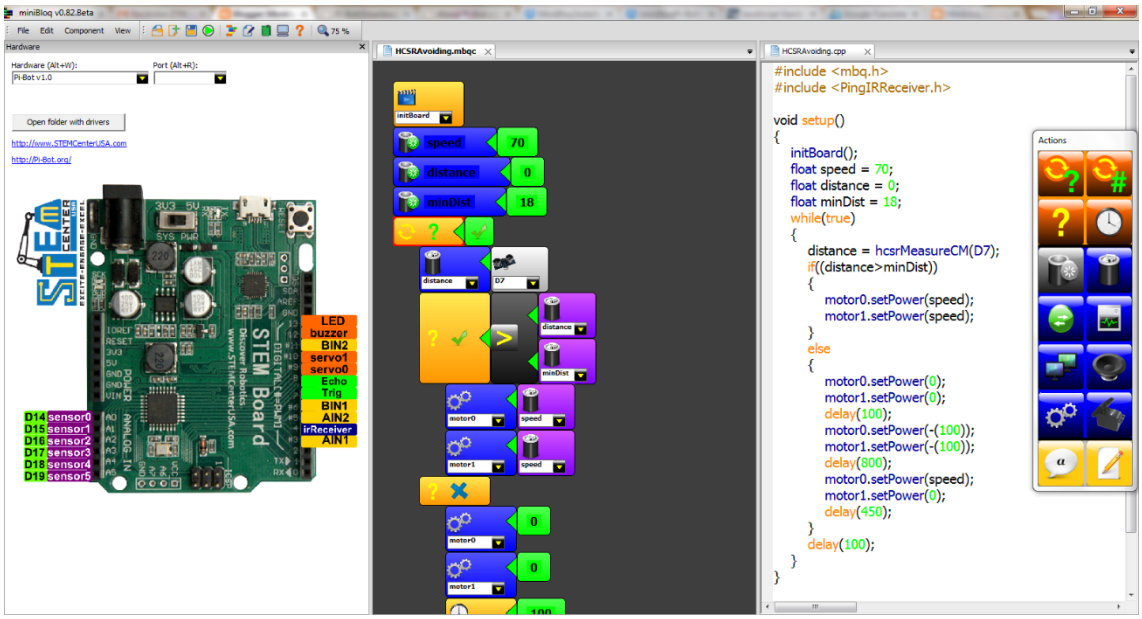

Ilustración 26. (2016). Ejemplo de uso Minibloq. Recuperado de www.blog.minibloq.org

Como se puede observar, la diferencia respecto a Ardublock radica en que, en este caso, se puede llevar a cabo paralelamente el diseño de los pines de la placa Arduino que se vaya a utilizar en la ventana situada a la izquierda. Por otro lado, en el resto de ventanas tiene lugar la implementación del comportamiento del programa.

Capítulo 3. Análisis y Diseño del sistema

En este capítulo, primero, se llevará a cabo un estudio de todos los requisitos que deberá cumplir el sistema. Además, se expondrán los diferentes casos de uso que pueden darse durante su utilización, con el fin de contemplar todas las posibles situaciones y la manera de actuar frente a ellas.

Más adelante tendrá lugar el diseño y la implementación final del sistema, donde se deberá de tener en cuenta todo el listado de requisitos para su realización. Para corroborar su correcta implantación, se harán todas las pruebas necesarias que acrediten el cumplimiento de todas las funciones necesarias.

3.1 Análisis

En este apartado tendrá lugar la enumeración de los requisitos de usuario que tienen lugar en este proyecto, además de los requisitos de software y los casos de uso.

La clasificación de los requisitos de usuario será la siguiente:

- Requisito de usuario de capacidad: requisito que especifica lo que puede realizar el software para satisfacer las necesidades del usuario.
- Requisito de usuario de restricción: requisito que especifica una restricción sobre la creación o un cierto comportamiento del software.

Para representar cada uno de los requisitos se seguirá el siguiente formato plasmado en la tabla 14:

Identificador: Rx-xx (nombre)	
Fuente:	
Prioridad:	☐ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☐ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☐**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	

Tabla 14. Formato de requisito de usuario

Siendo cada uno de sus campos:

- Identificador: nombre único y breve del requisito en cuestión. Para los requisitos de usuario se empleará la nomenclatura ‘RU’, mientras que para los de software será ‘RS’. Por otro lado, ‘xx’ representará el número de requisito, como por ejemplo, RU-02.
- Fuente: de donde proviene el requisito. Puede ser del cliente o del propio analista. Para el caso de los requisitos de Software se referenciará el requisito de Usuario del que proviene.
- Prioridad: parámetro que indica la prioridad de implementación de dicho requisito.
- Necesidad: parámetro que indica la necesidad de tener en cuenta dicho requisito a nivel de este proyecto.
- Claridad: parámetro que indica si el requisito está explicado de forma sencilla y entendible, impidiendo la existencia de ambigüedad en su contenido.
- Verificabilidad: parámetro que indica cómo de fácil se puede comprobar que dicho requisito ha sido implantado correctamente en el sistema.
- Estabilidad: parámetro que indica si dicho requisito puede ser modificado durante el resto de realización del proyecto.
- Descripción: descripción completa del requisito en cuestión atendiendo a las pautas para lograr una especificación libre de ambigüedades.

Análogamente, para el estudio de los casos de uso existentes, se tiene como base el esquema de posibilidades de acción que hay entre los dos actores, el usuario y el vehículo, siendo este el siguiente (ilustración 27):

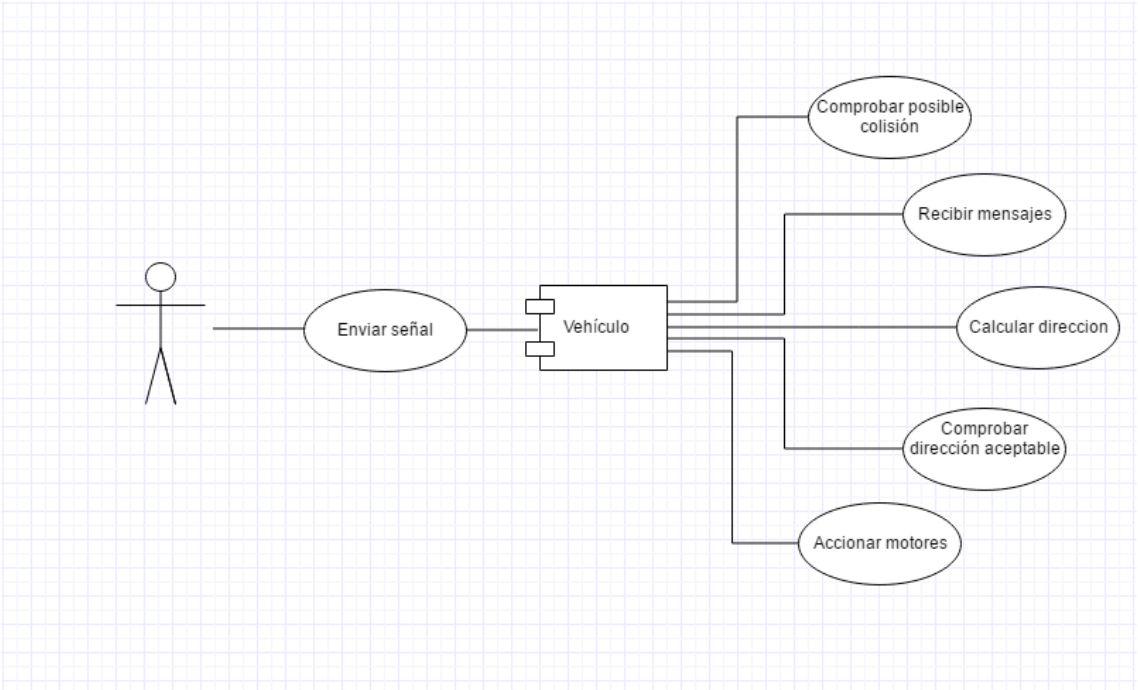

Ilustración 27. Imagen de casos de uso

Por una parte, el usuario únicamente tiene la funcionalidad de emitir la señal mediante el dispositivo emisor de frecuencia. Por la otra parte, el vehículo es, en esencia, el que tiene el mayor abanico de posibles acciones.

A partir de dicho esquema se obtuvieron distintos casos uso, los cuales se representan por medio de la tabla 15:

Identificador	**CU-xx**	
Nombre		
Actor	Usuario/vehículo	
Descripción		
Precondición		
Secuencia normal	Paso	Acción
Postcondición		
Excepción		

Tabla 15. Plantilla de casos de uso

Siendo cada uno de los campos:

- Identificador: nombre único del caso de uso en cuestión. 'xx' representa el número de caso de uso, por ejemplo, CU-02.
- Nombre: descripción abreviada y sin dar detalles del escenario
- Actor: el que ejerce el papel protagonista en dicho caso de uso. Puede ser el usuario o el vehículo.
- Descripción: descripción completa del escenario en cuestión lo más clara posible.
- Precondición: situación que tiene lugar antes del suceso.
- Secuencia normal: pasos que tienen lugar sin darse ninguna excepción de lo que sería una actuación normal en dicho escenario.
- Postcondición: la situación que debería darse al finalizar la secuencia de actuación.
- Excepción: comportamiento anómalo que puede darse durante la ejecución de la escena.

3.1.1 Requisitos de Usuario

Requisitos de usuario de capacidad

Identificador: RU-01 (Apagar motores vehículo)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El usuario deberá poder desconectar la alimentación de los motores para detener el funcionamiento de los mismos.

Tabla 16. Requisito de usuario RU-01

Identificador: RU-02 (Vehículo autónomo)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El usuario no podrá establecer los movimientos exactos que haga el vehículo. El software los seleccionará por sí mismo.

Tabla 17. Requisito de usuario RU-02

Identificador: RU-03 (Detención por proximidad vehículo-emisor)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Si la distancia entre el usuario y el vehículo es corta, el vehículo permanecerá inmóvil.

Tabla 18. Requisito de usuario RU-03

Identificador: RU-04 (Desconexión dispositivo emisor)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El usuario podrá desconectar el mando emisor si desea que el vehículo deje de detectarle.

Tabla 19. Requisito de usuario RU-04

Identificador: RU-05 (Tiempo de respuesta seguimiento)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El tiempo de respuesta para cada movimiento del vehículo debe ser menor a 2 segundos.

Tabla 20. Requisito de usuario RU-05

Identificador: RU-06 (Feedback vehículo)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El vehículo tiene que mostrar mediante uso de LEDs que está operativo, o si en su defecto tiene algún problema.

Tabla 21. Requisito de usuario RU-06

Identificador: RU-07 (Feedback dispositivo emisor)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El dispositivo emisor debe mostrar mediante un LED que está funcionando.

Tabla 22. Requisito de usuario RU-07

Identificador: RU-08 (Detección obstáculos)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El vehículo deberá poder detectar obstáculos delante de él con el fin de evitar colisiones.

Tabla 23. Requisito de usuario RU-08

Identificador: RU-09 (Sistema de motores)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Para proveer de movimiento al vehículo se dispondrá de un sistema de dos ruedas motorizadas.

Tabla 24. Requisito de usuario RU-09

Identificador: RU-10 (Triangulación emisor)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El software del vehículo deberá detectar por medio de triangulación por radiofrecuencia la posición del elemento emisor.

Tabla 25. Requisito de usuario RU-10

Identificador: RU-11 (Emisor fuera de alcance)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Si el usuario está a una distancia mayor que la soportada por los componentes establecedores de la comunicación por radiofrecuencia, al no llegar señal al vehículo éste permanecerá inmóvil.

Tabla 26. Requisito de usuario RU-11

Identificador: RU-12 (Velocidad vehículo)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El software del vehículo podrá cambiar la velocidad de los motores a la que sea adecuada.

Tabla 27. Requisito de usuario RU-12

Identificador: RU-13 (Alimentación placa Arduino)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Las dos placas empleadas deberán contar con un sistema de alimentación externa.

Tabla 28. Requisito de usuario RU-13

Identificador: RU-14 (Distancia mínima seguimiento)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El usuario deberá alejarse del vehículo una distancia mínima si desea que se mueva hacia él.

Tabla 29. Requisito de usuario RU-14

Identificador: RU-15 (Tolerancia a fallos)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se deberá implementar un sistema de fallos en caso de que el sistema calcule una dirección errónea.

Tabla 30. Requisito de usuario RU-15

Requisitos de usuario de restricción

Identificador: RU-16 (Distancia emisor-vehículo)	
Fuente/s:	**Cliente**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El usuario deberá permanecer hasta una distancia máxima del vehículo si quiere mantener la comunicación entre el emisor y los receptores.

Tabla 31. Requisito de usuario RU-16

Identificador: RU-17 (Entorno de desarrollo)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☒ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☐ **Esencial** ☒ **Deseable** ☐ **Opcional**
Descripción:	Se empleará el entorno de desarrollo propio de Arduino para la creación del software.

Tabla 32. Requisito de usuario RU-17

Identificador: RU-18 (Lenguaje de programación)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒ **Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El lenguaje empleado para la programación del software será Arduino.

Tabla 33. Requisito de usuario RU-18

Identificador: RU-19 (Número de placas)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒ **Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se emplearán 2 placas de Arduino mega, una para el emisor y otra para el conjunto de receptores.

Tabla 34. Requisito de usuario RU-19

Identificador: RU-20 (Instalación librerías adicionales)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Deberá contemplarse la existencia de librerías externas que puedan servir para un correcto uso de los componentes empleados.

Tabla 35. Requisito de usuario RU-20

Identificador: RU-21 (Voltaje placas Arduino)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El voltaje mínimo para la placa será de 6V y el máximo de 20V.

Tabla 36. Requisito de usuario RU-21

Identificador: RU-22 (Voltaje motores)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Los motores tendrán una fuente de alimentación propia, e independiente a la de las placas, con la suficiente corriente para que funcionen correctamente.

Tabla 37. Requisito de usuario RU-22

Identificador: RU-23 (Número de módulos receptores)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Estabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se dispondrán de 4 receptores para la tarea de triangulación del emisor.

Tabla 38. Requisito de usuario RU-23

Identificador: RU-24 (Número de módulos emisores)	
Fuente/s:	**Cliente**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El dispositivo emisor dispondrá de exactamente 1 módulo emisor.

Tabla 39. Requisito de usuario RU-24

3.1.2 Casos de uso

Usuario

Identificador	CU-01	
Nombre	Mando emisor fuera de alcance	
Actor	Usuario	
Descripción	El dispositivo emisor, al estar fuera de alcance respecto a los receptores del vehículo, no consigue que su señal llegue a ellos.	
Precondición	El usuario intenta que el vehículo le detecte sin resultados	
Secuencia normal	Paso	Acción
	1	Acercar el mando al vehículo hasta que muestre señales de detección correcta
Postcondición	El vehículo ha detectado al emisor e inicia el proceso de seguimiento.	
Excepción	-	

Tabla 40. Caso de uso CU-01

Identificador	CU-02	
Nombre	Emisión de señal correcta al vehículo	
Actor	usuario	
Descripción	El usuario empieza a enviar señal a una distancia aceptable respecto al vehículo.	
Precondición	El usuario se encuentra con el dispositivo emisor de señal apagado	
Secuencia normal	Paso	Acción
	1	El usuario enciende el dispositivo emisor y comienza a enviar señal
Postcondición	El vehículo vuelve a tener corriente para seguir moviéndose	
Excepción	-	

Tabla 41. Caso de uso CU-02

Vehículo

Identificador	CU-03	
Nombre	Señal detectada hacia una dirección	
Actor	Vehículo	
Descripción	De los 4 receptores, se obtiene mayor señal respecto a las otras en uno de ellos, por lo cual el vehículo deberá dirigirse hacia allí.	
Precondición	El usuario, mediante el dispositivo emisor, emite señales a los receptores estando en una cierta posición alejada, pero dentro de los límites.	
Secuencia normal	Paso	Acción
	1	Se obtiene la señal en los receptores
	2	Se calcula la dirección de donde proviene
	3	Se procede a mover el coche hacia dicha dirección
Postcondición	El vehículo se mueve en la dirección donde está el mando emisor	
Excepción	Por la probabilidad existente de fallo en el cálculo de la dirección, la dirección a la que se mueve el vehículo no es la ideal.	

Tabla 42. Caso de uso CU-03

Identificador	CU-04	
Nombre	Señal detectada en cada receptor por igual	
Actor	Vehículo	
Descripción	El usuario, mediante el dispositivo emisor, emite señales a los receptores estando próximo al vehículo, con el resultado de que a todos les llega la misma cantidad de comunicación por igual, por lo que el vehículo deberá quedarse inmóvil.	
Precondición	El usuario está al lado del vehículo.	
Secuencia normal	Paso	Acción
	1	Se obtiene la señal en los receptores
	2	Se obtiene prácticamente la misma cantidad de comunicación en todos ellos.
	3	El vehículo mantiene en el sitio
Postcondición	Misma escena que en la precondición	
Excepción	Hay algún fallo en la emisión de señal, por lo tanto el vehículo no detecta nada	

Tabla 43. Caso de uso CU-04

Identificador	CU-05	
Nombre	Obstáculo en el camino del vehículo	
Actor	vehículo	
Descripción	En el camino del vehículo al usuario hay un obstáculo	
Precondición	El vehículo circula en modo seguimiento	
Secuencia normal	Paso	Acción
	1	El vehículo detecta un obstáculo próximo a él
	2	El vehículo se detiene
	3	El vehículo comprueba la localización exacta del obstáculo
	4	El vehículo evita el obstáculo
	5	El vehículo vuelve a comprobar la existencia de obstáculos. En caso afirmativo, se repiten los pasos del 2-5. Si no hay obstáculo, vuelve a la tarea de seguimiento
Postcondición	El vehículo vuelve a la tarea se seguimiento	
Excepción	-	

Tabla 44. Caso de uso CU-05

Identificador	CU-06	
Nombre	Dirección calculada errónea	
Actor	vehículo	
Descripción	Por la pequeña probabilidad existente de que se reciba más comunicación en un receptor, el cual está situado más lejos del emisor que otro receptor, el software ha calculado una dirección incorrecta	
Precondición	El vehículo circula en modo seguimiento	
Secuencia normal	Paso	Acción
	1	El vehículo calcula la dirección errónea
	2	El vehículo comprueba la viabilidad de la dirección
	3	Al tratarse de una dirección errónea, el vehículo toma la dirección que se calculó en la iteración anterior
	4	El vehículo vuelve a calcular una nueva dirección, aceptando la dirección que supuestamente era errónea si se repite.
Postcondición	El vehículo continua con la tarea se seguimiento	
Excepción	-	

Tabla 45. Caso de uso CU-06

3.1.3 Requisitos software

Funcionales

Identificador: RS-01 (Apagar motores)	
Fuente/s:	**RU-01**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El sistema permitirá desconectar o encender la alimentación de los motores mediante un interruptor físico.

Tabla 46. Requisito de software RS-01

Identificador: RS-02 (Autonomía de movimientos del vehículo)	
Fuente/s:	**RU-02**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El sistema, internamente, será capaz de comprobar cuál es la dirección a tomar, ya sea bien quedándose quieto, activando el modo anti-colisión, o tomando el camino apropiado hasta el emisor mediante la triangulación de su posición.

Tabla 47. Requisito de software RS-02

Identificador: RS-03 (Detención por proximidad)	
Fuente/s:	**RU-03**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Si al triangular la posición, todos los receptores detectan por igual al emisor, se considerará que están próximos y el vehículo se detendrá.

Tabla 48. Requisito de software RS-03

Identificador: RS-04 (Desconexión dispositivo emisor)	
Fuente/s:	**RU-04**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El dispositivo emisor tendrá un interruptor físico para poder apagarlo y dejar de emitir señal.

Tabla 49. Requisito de software RS-04

Identificador: RS-05 (Tiempo de respuesta seguimiento)	
Fuente/s:	**RU-05**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Los procesos internos del software del vehículo para hacer moverse al vehículo no deben superar un intervalo de 2 segundos.

Tabla 50. Requisito de software RS-05

Identificador: RS-06 (Feedback detección emisor)	
Fuente/s:	**RU-06**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El vehículo mostrará mediante un LED encendido el momento en el que esté recibiendo señales y se apagará cuando acabe la tarea.

Tabla 51. Requisito de software RS-06

Identificador: RS-07 (Feedback activación sistema anti-colisión)	
Fuente/s:	**RU-06**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El vehículo mostrará mediante un LED encendido el intervalo en el que esté activado el modo anti-colisiones, y se apagará cuando acabe la tarea.

Tabla 52. Requisito de software RS-07

Identificador: RS-08 (Feedback envío de señal)	
Fuente/s:	**RU-07**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El dispositivo emisor dispondrá de un LED que permanecerá encendido mientras se suministre corriente al circuito, lo cual permitirá enviar la señal, y permanecerá apagado si no hay corriente, y por tanto no se envía señal.

Tabla 53. Requisito de software RS-08

Identificador: RS-09 (Sistema anti-colisión)	
Fuente/s:	**RU-08**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Si el vehículo se encuentra de frente con un obstáculo, comenzará el modo anti-colisión, el cual evitará el obstáculo, y una vez no haya ninguna obstrucción, volverá a ejecutarse con normalidad, esto es, aplicando la tarea de seguimiento.

Tabla 54. Requisito de software RS-09

Identificador: RS-10 (Sistema de motores)	
Fuente/s:	**RU-09**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El software del vehículo deberá estar adaptado para hacer funcionar un sistema de tracción de dos ruedas motorizadas.

Tabla 55. Requisito de software RS-10

Identificador: RS-11 (Triangulación emisor)	
Fuente/s:	**RU-10**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Para triangular la posición del emisor, se hará uso de 4 módulos receptores más uno emisor. Como el software de los módulos de frecuencia no asegura la comunicación, se hará uso de estadísticas para comprobar la posible posición del emisor, maximizando el número de resultados correctos mediante implantación de tolerancia a fallos.

Tabla 56. Requisito de software RS-11

Identificador: RS-12 (Emisor fuera de alcance)	
Fuente/s:	**RU-11**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Cuando el dispositivo emisor esté fuera del rango de alcance de la comunicación, y por tanto no se le detecte, se pararán los motores.

Tabla 57. Requisito de software RS-12

Identificador: RS-13 (Velocidad vehículo)	
Fuente/s:	**RU-12**
Prioridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	La velocidad global del vehículo en seguimiento tendrá de valor 180 para el controlador de los motores, ya que es la velocidad adecuada para un seguimiento óptimo, siendo la media de velocidad con la que

	camina una persona de 1,5m/s. En caso de activarse el modo anti-colisiones, la velocidad se incrementará hasta el máximo, 255, para poder volver a la tarea de seguimiento cuando antes sea posible.

Tabla 58. Requisito de software RS-13

Identificador: RS-14 (Alimentación placas Arduino)	
Fuente/s:	**RU-13**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se hará uso de un portapilas conectado a cada placa Arduino para suministrarles corriente.

Tabla 59. Requisito de software RS-14

Identificador: RS-15 (Distancia mínima seguimiento)	
Fuente/s:	**RU-14**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	En cuanto uno de los receptores tenga en mayor porcentaje el número de mensajes entrantes por parte del emisor, se considerará que el usuario está en dicha dirección, y se tomará la dirección correspondiente de seguimiento.

Tabla 60. Requisito de software RS-15

Identificador: RS-16 (Distancia máxima emisor-vehículo)	
Fuente/s:	**RU-16**
Prioridad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Existe una distancia máxima de aproximadamente dos metros, a partir de los cuales dejará de recibir señal el vehículo, y por tanto no podrá detectar al emisor.

Tabla 61. Requisito de software RS-16

Identificador: RS-17 (Voltaje placas Arduino)	
Fuente/s:	**RU-21**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se suministrará un total de 6V de corriente provenientes de un portapilas conectado a cada placa Arduino como fuente de alimentación externa.

Tabla 62. Requisito de software RS-17

Identificador: RS-18 (Voltaje motores)	
Fuente/s:	**RU-22**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El valor de voltaje para los motores tendrá el que permite el portapilas que se instalará en el chasis. Será de 6V, lo cual lo compondrán 4 pilas de 1.5V conectadas en serie.

Tabla 63. Requisito de software RS-18

Identificador: RS-19 (Número de módulos receptores)	
Fuente/s:	**RU-23**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Estabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Para la triangulación del emisor el vehículo dispondrá de exactamente 4 módulos receptores, localizados de manera que entre ellos las posiciones sean similares a los puntos cardinales (norte, sur, este, oeste).

Tabla 64. Requisito de software RS-19

Identificador: RS-20 (Optimización localización módulos receptores)	
Fuente/s:	**RU-23**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☐ **Alta** ☐ **Media** ☒ **Baja**
Estabilidad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Para optimizar la detección por parte de los módulos receptores, se tendrán que disponer de manera que se solapen en la comunicación lo menos posible, y sirva para que se localice con más precisión la dirección sobre la que está el emisor.

Tabla 65. Requisito de software RS-20

Identificador: RS-21 (Número de módulos emisores)	
Fuente/s:	**RU-24**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Únicamente se usará 1 módulo emisor para emitir señal a los receptores. No son necesarios más debido a que no hay diferentes

	mensajes ni otras opciones que se tengan que tener en cuenta por parte del emisor.

Tabla 66. Requisito de software RS-21

Identificador: RS-22 (Tolerancia a fallos)	
Fuente/s:	**RU-15**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☐ **Alta** ☒ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	El software deberá tener un sistema a prueba de fallos en el cálculo de la dirección. Si la dirección calculada en la iteración actual, es totalmente opuesta a la calculada anteriormente, se volverá a ejecutar la dirección anterior. En la siguiente iteración, si la dirección que parecía errónea persiste, se dará por válida y será la que se siga.

Tabla 67. Requisito de software RS-22

No funcionales

Identificador: RS-23 (Modelo de placa Arduino)	
Fuente/s:	**RU-19**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Las dos placas que se emplearán serán el modelo Arduino Mega 2560.

Tabla 68. Requisito de software RS-23

Identificador: RS-24 (Instalación librerías adicionales)	
Fuente/s:	**RU-20**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Para la creación del software deberán instalarse las librerías necesarias para su correcto funcionamiento: 1. VirtualWire para los componentes de radiofrecuencia. 2. NewPing para el uso del componente anti-colisiones.

Tabla 69. Requisito de software RS-24

Identificador: RS-25 (Entorno de desarrollo)	
Fuente/s:	**RU-17**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☐**Esencial** ☒ **Deseable** ☐ **Opcional**
Descripción:	El entorno de desarrollo para la creación del software será Arduino IDE.

Tabla 70. Requisito de software RS-25

Identificador: RS-26 (Lenguaje de programación)	
Fuente/s:	**RU18**
Prioridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Claridad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Verificabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Estabilidad:	☒ **Alta** ☐ **Media** ☐ **Baja**
Necesidad:	☒**Esencial** ☐ **Deseable** ☐ **Opcional**
Descripción:	Se usará el lenguaje propio de Arduino para confeccionar el software controlador del vehículo.

Tabla 71. Requisito de software RS-26

3.1.4 Matriz de trazabilidad de requisitos

	RU01	RU02	RU03	RU04	RU05	RU06	RU07	RU08	RU09	RU10	RU11	RU12	RU13	RU14	RU15	RU16	RU17	RU18	RU19	RU20	RU21	RU22	RU23	RU24
RS01	x																							
RS02		x																						
RS03			x																					
RS04				x																				
RS05					x																			
RS06						x																		
RS07						x																		
RS08							x																	
RS09								x																
RS10									x															
RS11										x														
RS12											x													
RS13												x												
RS14													x											
RS15														x										
RS16																x								
RS17																					x			
RS18																						x		
RS19																							x	
RS20																							x	
RS21																								x
RS22															x									
RS23																x			x					
RS24																				x				
RS25																	x							
RS26																		x						

Tabla 72. Matriz de trazabilidad de requisitos

3.2 Diseño

En esta parte del capítulo va a tener lugar el diseño tanto del hardware como del software necesario para el cumplimiento de este proyecto. Se hará especial hincapié en el estudio del funcionamiento de cada componente, para asegurar, y además dar a conocer, la correcta comprensión del material usado.

3.2.1 Diseño del hardware

En el apartado 2.3 ya se vio una introducción acerca de cada componente usado en el prototipo del vehículo. En esta ocasión el documento se centrará en desarrollar más a fondo cada uno de ellos, además de aportar el esquema de conexión entre la placa Arduino Mega con cada uno. Para dicho esquema de conexión se creará una tabla referenciando al pin del componente junto con el pin de la placa Arduino al que se conecta.

Únicamente, en las tablas se mostrarán los pines que implican el tráfico de datos, mientras que los que se utilizan meramente para proveer de corriente al componente, sólo serán visualizados en el esquema gráfico de cada conexión.

3.2.1.1 Chasis del vehículo

Como se pudo observar en la ilustración 13, este tipo de chasis cuenta con dos ruedas sobre las que se ejerce el movimiento, ayudadas de otra rueda llamada rueda loca. Esta rueda loca, únicamente sirve de apoyo con el fin de que les sea estable y menos costosa la tarea de ejercer el movimiento sobre el vehículo a los motores.

Además, se puede ver que salen dos cables distintos del portapilas. En el apartado de la motorización del vehículo, ubicado después de éste, se explicará para qué sirven.

Por otro lado, durante el proceso de montaje, es recomendable dejar conectado un cable a cada toma de corriente del motor antes de montarlos, tal y como se muestra en la imagen 28:

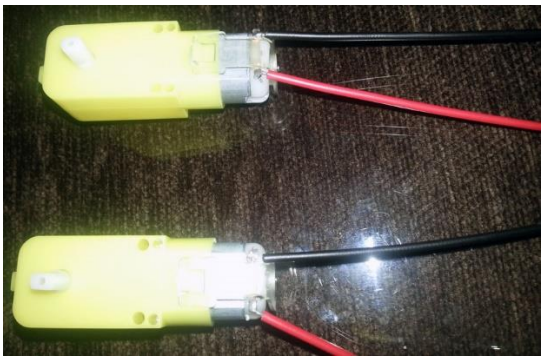

Ilustración 28. Isaac PE. (2014) Cableado motores. Recuperado de www.comohacer.eu

Es de vital importancia respetar el orden de colocación de los pares de cables de colores sobre los pares de conexiones de cada motor como se revela en la ilustración 28, ya que así se evitarán conexiones erróneas en el circuito final, dándose el incidente de que alguno de los motores gire en el sentido contrario respecto al otro.

En este caso, se ha decidido soldar los cables para evitar problemas posibles tales como la desconexión de algún cable con el motor, lo cual haría que se tuviesen que realizar movimientos arriesgados sobre el circuito del vehículo, ya que habría que dar la vuelta al chasis para poder operar.

Finalmente, una vez se ha seguido el manual de montaje nos queda un chasis tal como el mostrado en las imágenes 29 y 30:

Ilustración 29. (2015). Chasis montado visto desde arriba. Recuperado de www.amazon.com

Ilustración 30. (2015). Chasis montado visto por debajo. Recuperado de www.amazon.com

3.2.1.2 Motorización del vehículo

Lo primero, y más complejo en esta sección, es el funcionamiento del módulo controlador de motores L298N.

Este dispositivo permite controlar el funcionamiento de hasta dos motores a la vez. Para ello, hace uso de lo que es llamado "Puente H", contando con dos de ellos en su interior.

Puente H

El Puente H es un circuito electrónico diseñado para poder hacer girar a un motor en ambos sentidos. Se pueden construir con componentes individuales, o utilizarlos directamente como circuitos integrados (un único componente).

En el caso de este proyecto, cada puente H consta de 4 interruptores, los cuales, dependiendo de los que estén activados, provocarán una corriente positiva o negativa sobre el motor, lo cual provocará que gire hacia delante o a la inversa respectivamente. En la imagen 31 se tiene la ilustración del circuito que comprende el puente H.

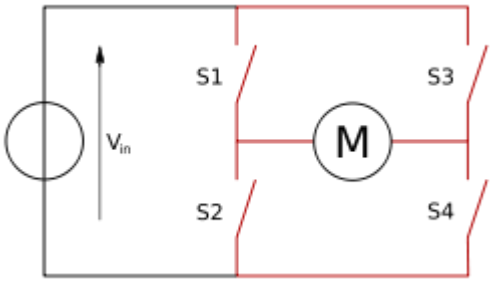

Ilustración 31. (2012). Puente H. Recuperado de www.blogspot.com

Como se puede observar, aparece la dirección hacia la que circula la corriente desde el origen. A partir de ahí, se podrían dar los siguientes casos recogidos en la tabla 73:

S1	S2	S3	S4	Acción
1	0	0	1	Giro hacia adelante
0	1	1	0	Giro hacia atrás
0	0	0	0	Freno por inercia
1	0	1	0	Freno brusco

Tabla 73. Combinaciones puente H

Una vez explicado el funcionamiento interno para los motores del controlador, llega el momento de proceder con el circuito que tendrá lugar a partir de él.

Anteriormente, se indicó que en cada motor había que conectar un cable a cada toma de corriente del propio motor. Por motivos de simplicidad, se recomendó emplear los mismos dos colores en cada par de cables, y conectar cada uno del mismo color sobre la toma de corriente correspondiente, de forma que cada motor quedara conectado igual que el otro, tal como quedaba reflejado en la ilustración 28.

Una vez se tiene hecho este paso, es el momento de ver dónde irá conectado cada motor al controlador de motores, además del resto de conexiones que tiene el módulo (imagen 32).

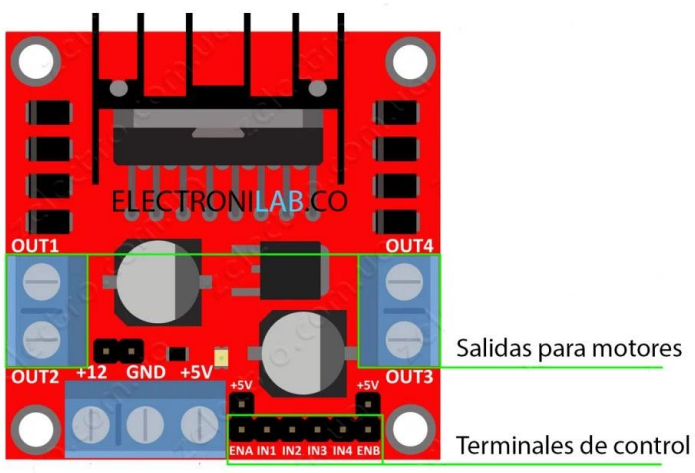

Ilustración 32. (2015). Salidas para motores del controlador L298N. Recuperado de www.thingsandcode.com

Como se puede observar, por una parte se tienen las salidas para los motores. Componen en total cuatro salidas, siendo dos para cada motor. Al aplicar el método ya indicado sobre el uso de pares de color de cables que van desde cada motor al controlador, quedaría el esquema mostrado en la ilustración 33:

Ilustración 33. (2015). Esquema conexión motores a controlador L298N. Recuperado de www.mercadolibre.com

Así, se ha conseguido evitar errores en cuanto a que alguno de los motores funcione al revés que el otro, provocando que sea muy complicada la tarea de sincronización de los motores en cuanto al movimiento que se desee ejercer sobre ellos.

Por otra parte, se tienen las salidas de control. Estas conexiones son las que van a la placa Arduino, y son las que establecen la comunicación entre los motores y el software, de manera que a través de ellas se puedan manejar los motores en cuanto al movimiento que se quiere que haga sobre cada uno, además de la velocidad de giro. Un ejemplo de su esquema de conexión es el que se puede ver a continuación, en la imagen 34:

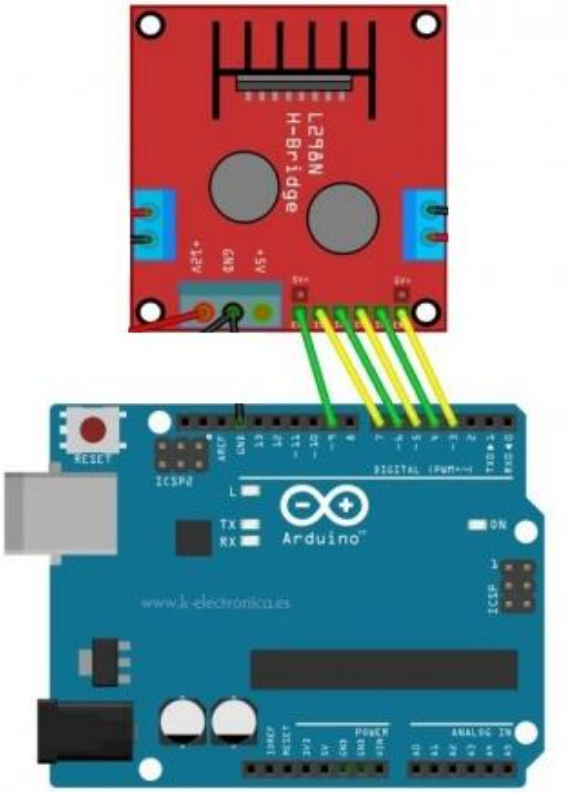

Ilustración 34. (2014). Esquema de conexión controlador L298N con Arduino. Recuperado de www.cxem.net

Como se puede ver, cada una de las seis salidas se conecta a un pin digital de la placa Arduino. En nuestro caso, gracias a los pines que provee la placa Arduino Mega, las conexiones de pines serán las mostradas en la tabla 74:

Pin módulo controlador	Pin placa Arduino Mega
ENA	38
IN1	13
IN2	12
IN3	11
IN4	10
ENB	39

Tabla 74. Conexión de pines controlador de motores

Para la velocidad de giro se utilizarán las conexiones de los extremos, llamadas ENA y ENB, con valores entre 0 siendo nula la velocidad y 255 siendo ésta la máxima. Respectivamente, la primera sirve para el motor A y la segunda para el motor B.

En cuanto a la dirección de giro de los motores, se usarán el resto de pines, llamados IN1 e IN2 para el motor A, y por otro lado IN3 e IN4 para el motor B. A continuación se muestra la tabla 75, la cual indica las dos posibilidades que existen, y los valores que se necesitan en los pines del motor:

Movimiento		
Girar hacia delante	Pin 1	HIGH
	Pin 2	LOW
Girar hacia atrás	Pin 1	LOW
	Pin 2	HIGH

Tabla 75. Parámetros para girar los motores

El valor HIGH sirve para activar dicha salida, mientras que el valor LOW es para desactivarla.

Una vez se tienen estos conocimientos, para el sistema de dos motores se pueden elaborar combinaciones para lograr diferentes resultados. Cada uno de ellos se expone seguidamente en la tabla 76:

Movimiento	Motor	Pin	Valor
Mover hacia delante	Motor izquierdo	IN1	HIGH
		IN2	LOW
	Motor derecho	IN3	HIGH
		IN4	LOW
Mover hacia atrás	Motor izquierdo	IN1	LOW
		IN2	HIGH
	Motor derecho	IN3	LOW
		IN4	HIGH
Girar a la derecha	Motor izquierdo	IN1	LOW
		IN2	HIGH
	Motor derecho	IN3	HIGH
		IN4	LOW
Girar a la izquierda	Motor izquierdo	IN1	HIGH
		IN2	LOW
	Motor derecho	IN3	LOW
		IN4	HIGH
Detener motores	Motor izquierdo	IN1	LOW
		IN2	LOW
	Motor derecho	IN3	LOW
		IN4	LOW

Tabla 76. Combinaciones para el movimiento del vehículo

Por último, quedan por exponer las conexiones que suministran la alimentación al componente, mostradas en la imagen 35:

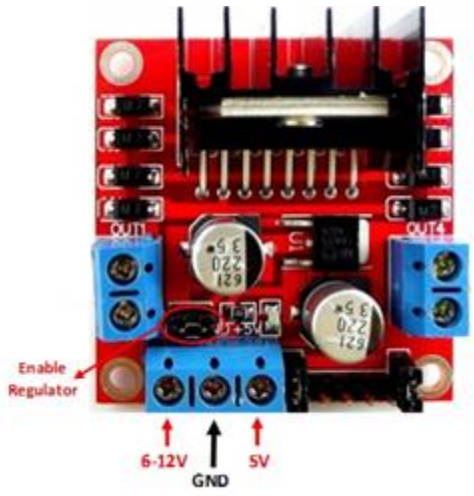

Ilustración 35. (2014). Pines de alimentación controlador L298N. Recuperado de www.electronicab.com

Por un lado, se tiene el jumper regulador (Enable Regulator), el cual al estar activado provoca que el propio componente utilice su regulador de energía para, a partir de ella, alimentar a la parte lógica del componente. Si se desactivara el jumper, sería necesario usar la conexión de la derecha de 5V para proveer de energía extra a la parte

lógica del módulo, ya que el regulador, al no funcionar, no proveerá de energía a dicha sección.

En este caso, el jumper permanecerá habilitado, para así poder alimentar el módulo con un voltaje de 6 a 12V sin tener que utilizar la conexión de 5V para suministrar energía extra a la parte lógica del módulo.

Una vez seleccionado éste método, sólo queda conectar la salida de 6-12V al lado positivo del portapilas instalado en el chasis, el cual aportará 6V, y finalmente conectar la salida GND a, por una parte, el lado negativo del portapilas, y por otro a la placa Arduino, quedando finalmente el esquema de la ilustración 36:

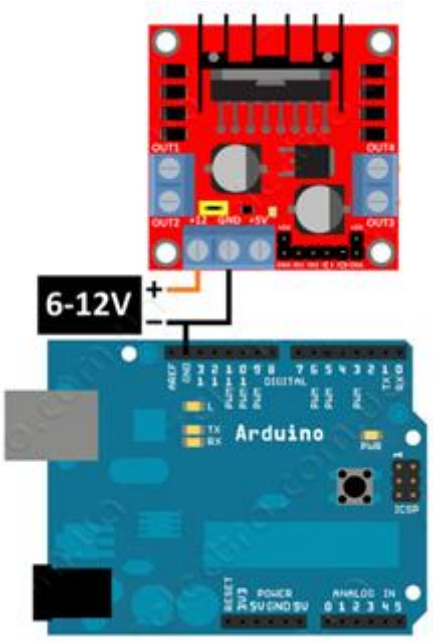

Ilustración 36. (2014). Esquema conexión alimentación controlador L298N. Recuperado de www.electronicab.com

3.2.1.3 Triangulación mediante comunicación por radiofrecuencia

En este apartado se va a explicar el diseño de la parte fundamental de este proyecto: la triangulación mediante comunicación por radiofrecuencia.

Para comenzar, primero se van a explicar los actores principales en esta tarea, los módulos RF utilizados.

Kit RF

Este par de módulos tienen como objetivo principal el paso de mensajes entre ellos, siendo siempre esta tarea de manera unidireccional.

Este paso de mensajes se lleva a cabo mediante lo llamado modulación ASK (Amplitude-shift keying). Esta modulación consiste en la conversión de datos digitales en variaciones de amplitud de onda según los datos enviados. Se lleva a cabo en el emisor, el cual se encarga de emitir las ondas que acaban en un módulo receptor, donde se vuelven a transformar a su valor en digital, obteniendo así el mensaje que fue enviado.

Una vez se ha estudiado la manera en la que estos componentes efectúan la comunicación entre ellos, toca establecer cómo se logrará triangular la posición del emisor mediante su mecánica de uso.

Uno de los principales problemas, y a la vez lo que hace más complejo el uso de estos sistemas para procesos como la triangulación, es el hecho de que no se puede obtener la intensidad con la que están comunicados este par de componentes, ni tampoco se puede obtener directamente una aproximación de la distancia a la que están.

Por ello, se hizo un estudio de las posibles soluciones a dicha cuestión mediante el uso de estos módulos:

1. Sistema de mallado: método por el cual se procedería a mallar el escenario con emisores, estableciendo un tipo de sistema de coordenadas, en el cual cada emisor emitiese un mensaje único para el receptor instalado en el vehículo, y así saber en qué zona del espacio está según los mensajes que le lleguen.
2. Sistema multireceptor: método en el cual se instalarían varios receptores en el vehículo colocados equitativamente, con el fin de triangular la posición del emisor dependiendo del receptor por el que mayor comunicación se haya producido.

El primer método se desestimó principalmente debido a que se necesita un gran espacio de trabajo para poder practicar este tipo de sistema, y no se dispone de ello.

Por lo tanto, se seleccionó el segundo, ya que además presentaba una mayor optimización en cuanto al uso del espacio y una disminución considerable en cuanto a costes.

Aplicando la opción seleccionada, el grupo de componentes lo formarán cuatro receptores y un emisor. El emisor, una vez encendido, no parará de enviar señales, las cuales podrá ser leídas por los receptores si están dentro del rango.

Por parte de los receptores, su localización será la siguiente que queda mostrada en la ilustración 37:

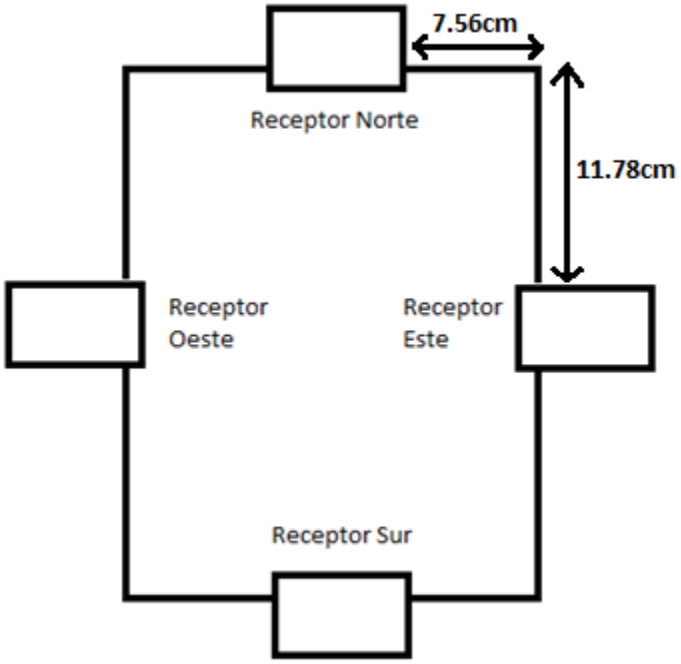

Ilustración 37. Localización receptores de frecuencia

Se ha establecido esta distribución ya que cubre de manera efectiva las diferentes áreas de detección del emisor, con el menor solapamiento posible entre las comunicaciones. Para mejor efectividad se necesitaría una mayor distancia entre los receptores, de manera que la comunicación entre cada uno de ellos con el emisor fuese más aislada, y por tanto si a alguno de ellos le llegasen mensajes sería mucho más probable que el emisor estuviese en dicha dirección. El inconveniente que hay para este aspecto es el pequeño tamaño del vehículo en cuestión, lo cual impide un mayor aislamiento de los receptores.

A continuación, se presenta una imagen de ejemplo (ilustración 38) de cómo sería el reparto ideal del área de detección entre todos los receptores:

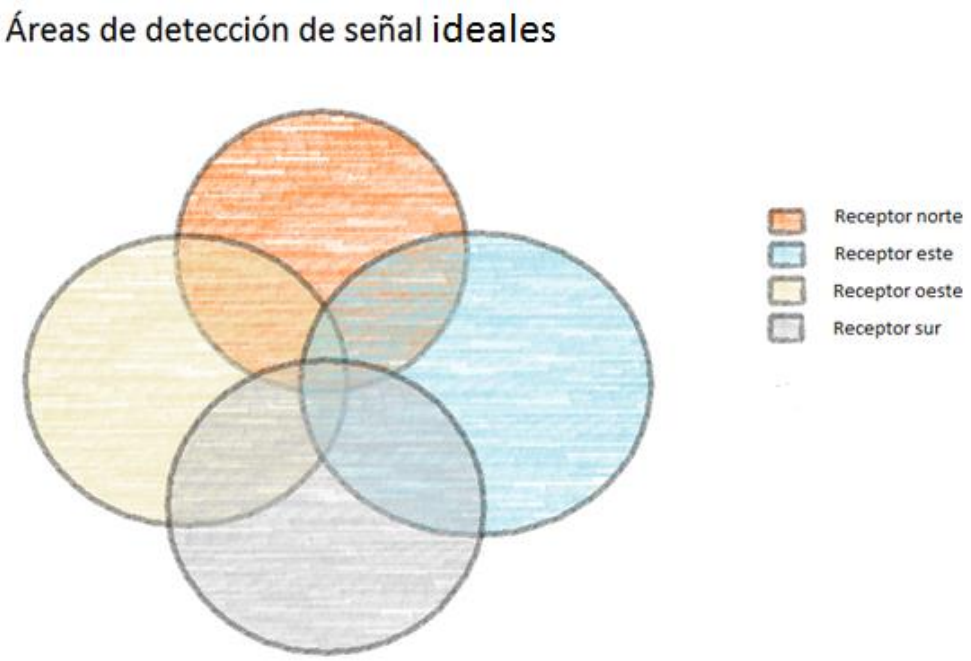

Ilustración 38. Área de impacto ideal de cada receptor

Como se puede observar, la idea es que haya zonas del espacio donde tenga mayor impacto cada receptor individualmente; a las cuales (al tener al emisor más cerca), lleguen la mayor parte de los mensajes, y por tanto se le detecte en esa dirección.

Para los demás casos, habrá que tener en cuenta que, dependiendo de la posición del emisor, puede estar en una zona en la que le detecten múltiples receptores. En este caso, se aplicarán las siguientes pautas:

- En caso de que todos los receptores lo detecten por igual, se considerará que el vehículo está al lado del emisor, y por tanto se detendrá.
- En caso contrario, es decir, haya uno o más receptores que ocupen prácticamente toda la detección de la señal, significará que dicha señal proviene de la dirección de alguno de ellos, tomando como dirección óptima la del receptor que más cantidad de comunicación haya tenido. En caso de empate, cualquiera sería viable.
- En caso de que ningún receptor reciba mensaje, el vehículo permanecerá inmóvil.

Las asociaciones de pines tanto para el emisor como los receptores con la placa Arduino son mostrados en las tablas 77 y 78:

Pin emisor	Pin placa Arduino Mega
Data	12

Tabla 77. Conexión pin emisor con Arduino

Pin receptores	Pin placa Arduino Mega
Receptor Norte - Data	4
Receptor Este - Data	5
Receptor Oeste - Data	6
Receptor Sur - Data	7

Tabla 78. Conexión pin de cada receptor con Arduino

Finalmente, queda por mostrar la visualización de dicha conexión entre cada uno de los componentes utilizados con la placa Arduino.

Por parte del emisor, se tiene en total 3 pines de conexión ilustrados en la imagen 39:

Ilustración 39. (2014). Pines de conexión del emisor. Recuperado de www.josehervas.es

- Data: pin por el cual se establecen los mensajes que son enviados por medio del software.
- VCC (5V): pin que suministra corriente al módulo.
- GND: toma de tierra.

Su esquema de conexión queda plasmado en la imagen 40:

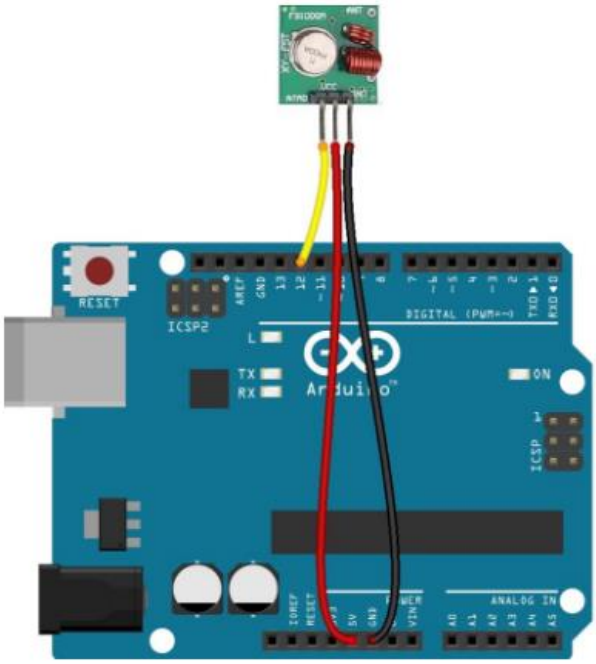

Ilustración 40. Xavier. Esquema de conexión del emisor. Recuperado de www.studioseed.com

En el caso de los receptores, hay en total 4 pines de conexión (imagen 41):

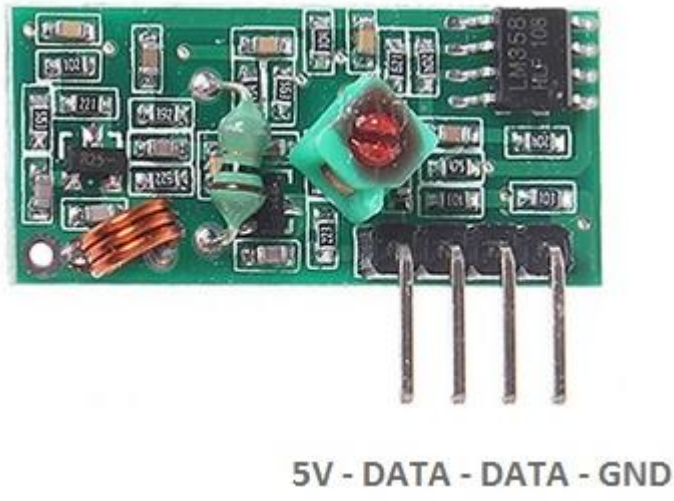

Ilustración 41. (2014). Pines de conexión de cada receptor. Recuperado de www.josehervas.es

- VCC (5V): pin que suministra corriente al módulo.
- 2 pines Data: pines por los cuales se tratan los mensajes recibidos por medio del software.
- GND: toma de tierra.

Siendo la ilustración 42 el sistema de conexión de cada uno de ellos:

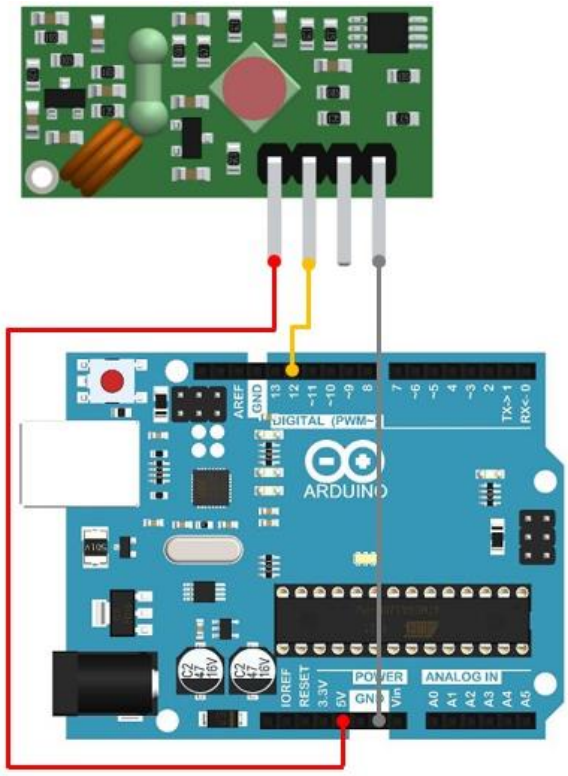

Ilustración 42. (2014). Esquema individual de conexión de receptor. Recuperado de www.cxem.net

Tan sólo hará falta conectar cada pin de datos de cada receptor de la manera que se reflejó en la tabla 74.

3.2.1.4 Sistema anti-colisiones

Para evitar choques frontales del vehículo con posibles obstáculos existentes en su camino hacia el emisor, se hará uso del módulo HC-SR04.

El principal funcionamiento de este componente se basa en medir distancias emitiendo ultrasonidos, que, tras impactar con un objeto, rebotan volviendo de nuevo al sensor y aportando el valor que necesita para calcular la distancia a la que está dicho objeto. Esto se lleva a cabo gracias a sus dos componentes con forma de cilindro, de los cuales uno de ellos se encarga de emitir los pulsos, y el otro de recibirlos.

El cálculo de la distancia detectada se llevará a cabo mediante el siguiente proceso:

1. Se cuenta con que la velocidad del sonido es de 343 m/s en condiciones de 20ºC de temperatura, 50% de humedad, y presión atmosférica a nivel del mar en el ambiente. Como este sensor trabaja con centímetros y microsegundos, haciendo la transformación el valor de la velocidad queda en 1/29.2 centímetros por microsegundo.

2. Durante el trayecto hasta que choca el pulso con un objeto, y vuelve, han pasado ‘t’ microsegundos.
3. Como el valor de tiempo que se tiene es el equivalente a la ida más la vuelta, para calcular la distancia real se tiene que usar la mitad, ‘t/2’, ya que es el tiempo real que tardó en impactar.
4. Ahora, según la fórmula que implica a la velocidad, la distancia y el tiempo, despejando la distancia nos queda el siguiente resultado:
    - distancia = velocidad * tiempo = 29.2 * t/2 cm.

La localización del componente para mejorar la fiabilidad será en la parte frontal del vehículo, cerca del suelo. Además, permanecerá fijo para que no haya desviaciones de la dirección de detección.

Por otro lado, para su integración con el sistema electrónico se seguirá el siguiente proceso:

Analizando el componente, el sensor posee 4 pines de conexión, ilustrados en la imagen 43:

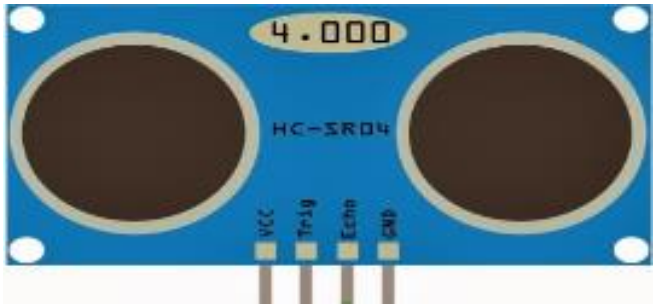

Ilustración 43. (2016). Pines de conexión sensor HC-SR04. Recuperado de www.sparkfun.com

- VCC: pin utilizado para suministrar voltaje al módulo.
- Trig: pin que corresponde a la creación del pulso.
- Echo: pin que corresponde a la recepción del pulso.
- GND: toma de tierra.

La tabla 79 recoge la conexión del componente con la placa:

Pin sensor ultrasónico	Pin placa Arduino Mega
Trig	2
Echo	3

Tabla 79. Conexión pines sensor ultrasónico con la placa Arduino

Por tanto, el esquema de conexión será el siguiente (ilustración 44):

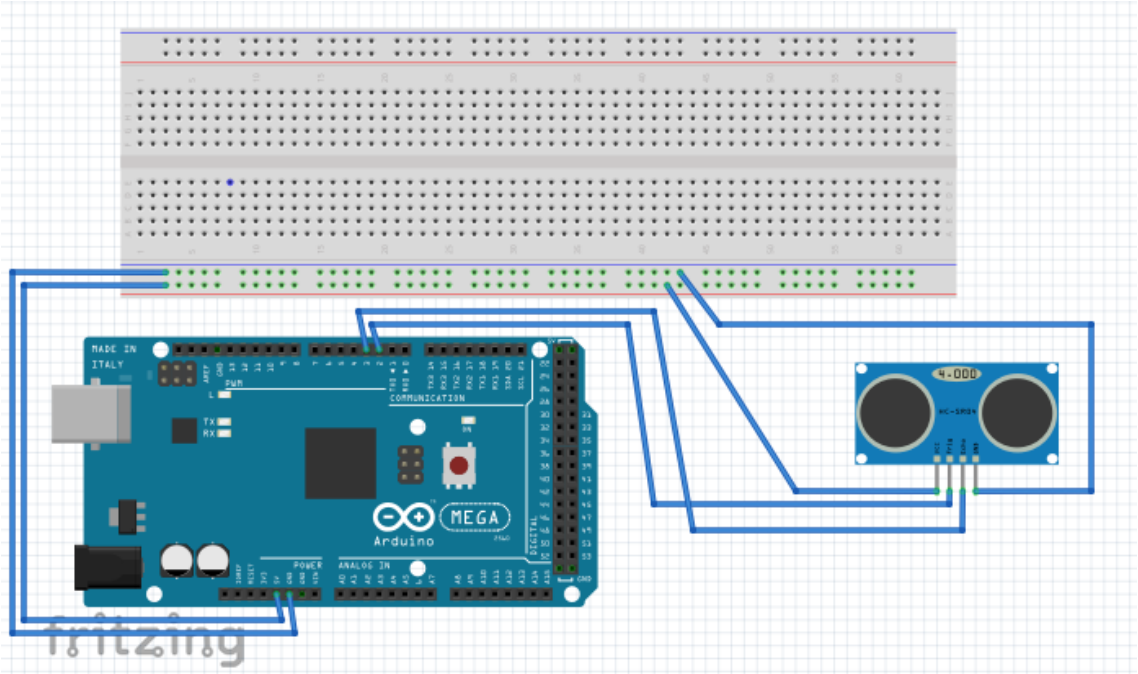

Ilustración 44. Esquema de conexión sensor HC-SR04

3.2.1.5 LEDs de retroalimentación

Lo último que queda por incluir son las retroalimentaciones solicitadas por medio de los requisitos.

En el caso del vehículo, se conectarán dos LEDs: uno indicará que el vehículo está ejecutando la tarea de detección de la señal proveniente del emisor, y otro señalizará que está activo el sistema anti-colisiones.

Cada LED necesita su correspondiente resistencia para evitar que se queme internamente. Además, su posición de colocación tiene que ser de manera que la pata que está completamente recta, sea la que se conecte a la resistencia.

El pin al que se conecta cada LED queda reflejado en la tabla 80.

Pin LED	Pin placa Arduino Mega
Led sistema seguimiento	22
Led sistema de colisión	24

Tabla 80. Conexión pines LEDs del vehículo

El esquema resultante es el siguiente que aparece en la imagen 45:

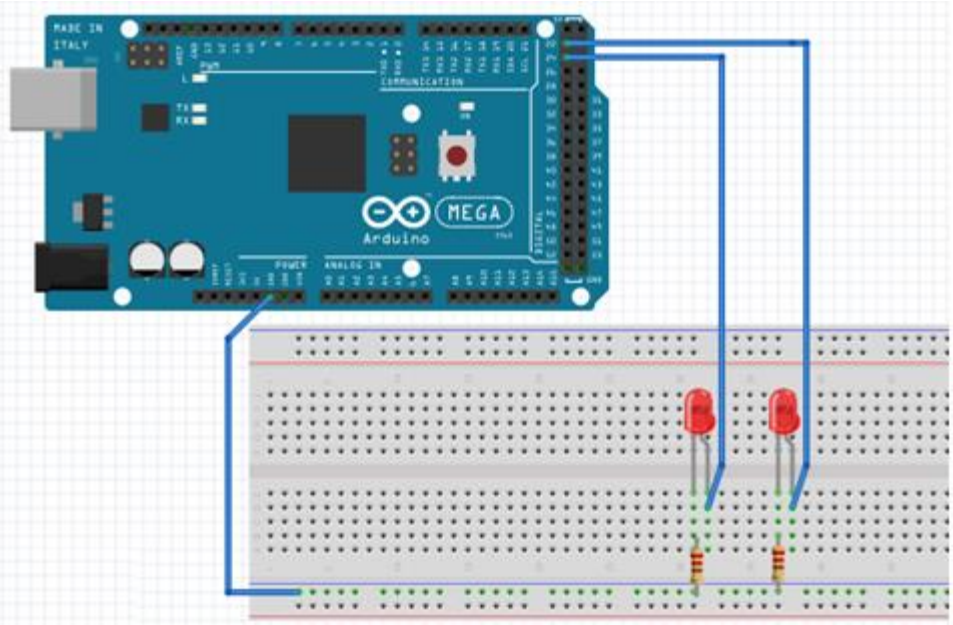

Ilustración 45. Esquema de conexión LEDs del vehículo

Independientemente, para el caso del emisor se conectará un LED que permanecerá encendido mientras se esté emitiendo señal.

El pin al que irá conectado será el siguiente, recogido en la tabla 81:

Pin LED	Pin placa Arduino Mega
Led emisor	13

Tabla 81. Conexión pin LED dispositivo emisor

El esquema final se muestra a continuación en la ilustración 46:

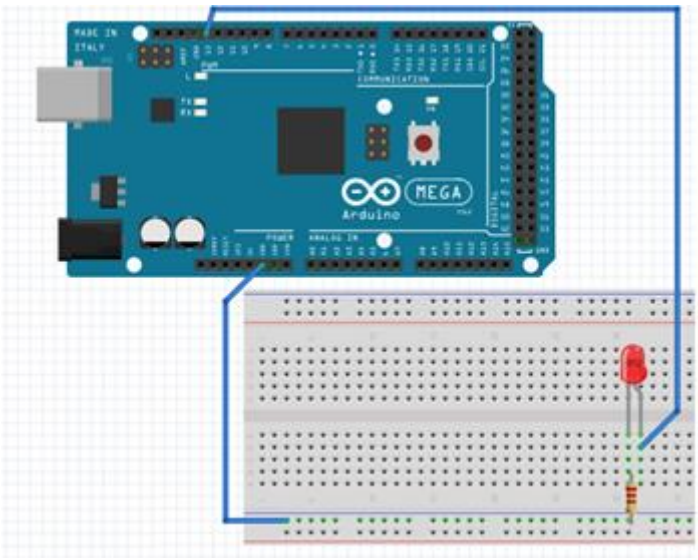

Ilustración 46. Esquema de conexión LED del emisor

3.2.2 Diseño del software

Una vez se ha diseñado el esquema de montaje del hardware que servirá para llevar a cabo las funciones en el sistema, es el momento de diseñarlas a nivel de programación.

Se sabe que todo programa en Arduino tiene dos métodos principales llamados setup() y loop(), explicados en el apartado 2.2.2. En el setup() tendrá lugar el establecimiento y la configuración de cada pin de la placa que tiene lugar en nuestro circuito, mientras que en el loop() tendrá lugar la ejecución de las instrucciones que marcarán el funcionamiento del controlador.

En este caso, además de estos dos métodos, por defecto se ha de incluir otros aspectos igual de importantes de cara a la obtención de una estructura clara y concisa, tales como el establecimiento de los parámetros globales del programa y la inclusión de las librerías que se necesiten, además de métodos adicionales, los cuales organizarán el código de manera que sea fácilmente localizable cualquier función disponible en el programa, mejorando además la legibilidad del código. Estos métodos serán llamados entre ellos, o directamente en el método loop(), dando como resultado el comportamiento del software.

Como resultado, en cada sketch quedará la siguiente estructura:

1. Importación de librerías necesarias
2. Establecimiento de parámetros globales
3. Métodos adicionales
4. Método setup()
5. Método loop()

Tanto el método setup() como el loop() irán en las últimas posiciones debido a que su funcionamiento consiste en el empleo de las funcionalidades implementadas.

3.2.2.1 Software del dispositivo emisor

Este programa será mucho más sencillo que en el caso del vehículo. Aquí, únicamente tiene lugar un módulo emisor de frecuencia conectado a la placa Arduino, por lo tanto sólo habrá que implementar un método que haga el envío de señales a través del módulo, el cual será llamado en el loop() para que se repita infinitas veces. Como funcionalidad adicional se tiene el uso de un LED que indica si está funcionando o no,

el cual se encenderá directamente en el setup(), ya que será suficiente con encenderlo al inicio del programa. El LED se apagará cuando se deje de suministrar energía.

La estructura final será la siguiente:

1. Importación de librerías necesarias
2. Establecimiento de parámetros globales
3. Método send()
4. Método setup()
5. Método loop()

3.2.2.2 Software del vehículo

Antes de establecer la estructura del software, se ha de tener en cuenta las funciones que van a tener lugar.

Las funciones aportadas por el hardware son las siguientes:

- Sistema anti-colisión
- Comunicación por radiofrecuencia
- Uso de motores para mover el vehículo
- Retroalimentación

Por orden de prioridad, es de vital importancia que el sistema compruebe primero la existencia de obstáculos. De nada sirve calcular la dirección del emisor si se va a producir un choque, ya que, al activarse el modo anti-colisiones, el vehículo quizá tome un camino totalmente diferente a la localización del usuario. Por lo tanto, en primer lugar irá la función anti-colisión, llamada comprobar_colision().

Después, en caso de que no haya obstáculos, se ha de empezar a detectar al emisor. Ésta es la función que conlleva la comunicación por radiofrecuencia, llamada recibir_mensajes(), la cual servirá para hacer un recuento de cuántos mensajes recibió cada uno de los receptores, y a su vez se usará, más adelante, para estimar desde dónde proviene la señal.

Se puede observar una ejemplificación de este proceso en la ilustración 47:

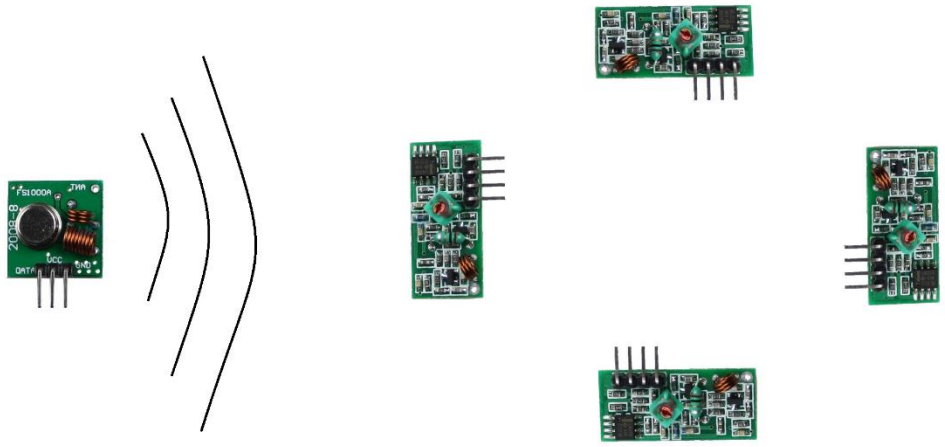

Ilustración 47. Ejemplo de envío de señal

Como mejora para esta tarea de comunicación, se ha establecido un mensaje específico para verificar la autenticidad de cada uno de ellos. El mensaje enviado tendrá el valor “M3nS4J3”, y los receptores tendrán que verificar la existencia de dicha cadena en el mensaje entrante, para comprobar que pertenece al emisor auténtico. Esta mejora tiene lugar debido a que podría aparecer otro dispositivo emisor de la misma frecuencia, y suponer un descontrol total en la comunicación. Se creará un método para la verificación del mensaje llamado comprobar_mensaje().

Una vez ha terminado la detección, habrá que crear una nueva función que, recopilando los datos, estime la dirección por donde, en un principio, se encuentra la localización del emisor. Ésta función será llamada calculo_direccion(). Si el vehículo detecta al emisor por el receptor sur, girará en el sentido de las agujas del reloj para evitar que se dé el caso de que se mantenga persiguiendo al emisor marcha atrás.

Por otro lado, debido al hecho de trabajar con estos módulos, la comunicación entre emisor-receptor no está asegurada, y puede darse el caso de que la dirección estimada sea totalmente contraria a la localización real del usuario. Por ello deberá crearse una funcionalidad adicional a prueba de fallos para este aspecto, llamada comprobar_direccion().

Esta funcionalidad comprobará si la dirección obtenida entra dentro de la lógica respecto a la anterior. Es decir, si se da el caso de que la dirección que se acaba de calcular es derecha, siendo la anterior izquierda, es muy probable que haya habido algún error en la transmisión de los mensajes. En el caso de que ocurra este suceso, se tomará

la dirección del paso anterior. Si vuelve a detectarse la misma dirección en la siguiente iteración, que supuestamente era errónea, se tomará por válida y se ejecutará.

Finalmente, una vez se tiene la dirección calculada y comprobada, sólo queda accionar los motores hacia la misma, y volver a iniciar la secuencia. El método se llamará accionar_motores().

La retroalimentación irá integrada en los métodos correspondientes. Consistirá en, por un lado, encender el LED de la comunicación cuando haya detectado un mínimo de 1 mensaje, y apagarlo cuando termine la tarea de recibir mensajes. Por otro lado, el segundo LED se encenderá cuando se activen los motores y se apagará cuando se detengan (siempre y cuando la orden provenga de la tarea de seguimiento).

Finalmente, reuniendo todos los requisitos anteriores, se obtiene el siguiente orden de ejecución (mostrado en la ilustración 48), incluyendo los métodos adicionales obtenidos:

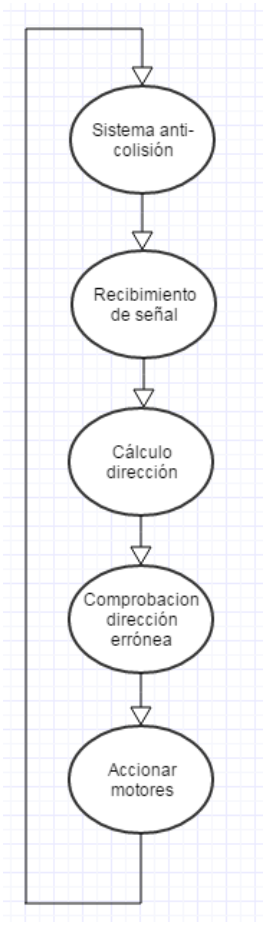

Ilustración 48. Orden de ejecución de tareas del software del vehículo

Siendo el diseño completo del sketch:

1. Importación de librerías necesarias
2. Establecimiento de parámetros globales
3. Método comprobar_colision()
4. Método recibir_mensajes()
5. Método calcular_direccion()
6. Método comprobar_fallo()
7. Método accionar_motores()
8. Método setup()
9. Método loop()

Para la retroalimentación, se tienen las siguientes pautas de diseño:

- El primer LED permanecerá encendido cuando esté operativa la función recibir_mensajes()
- El segundo LED se encenderá cuando, después de comprobar la existencia de obstáculos, se active el modo anti-colisiones al dar positiva la comprobación. Se apagará al salirse del modo.

Capítulo 4. Implantación y Pruebas del sistema

En esta sección del capítulo, va a tener lugar la descripción del código implementado tanto para el dispositivo emisor como para el software del vehículo.

Como se vio anteriormente, cada programa en Arduino sigue el siguiente esquema.

1. Importación de librerías necesarias
2. Establecimiento de parámetros globales
3. Métodos adicionales
4. Método setup()
5. Método loop()

Ahora, teniendo en cuenta esto, sólo queda detallar cada parte por separado.

4.1 Implantación del software del emisor

Librerías

Será necesario importar la librería VirtualWire, ya que es la librería que aporta las funciones de uso para el kit de radiofrecuencia de este proyecto (ilustración 49).

```
#include <VirtualWire.h>
```

Ilustración 49. Librerías sistema emisor

Parámetros globales

El único parámetro necesario ha sido el del valor del pin del LED usado para mostrar que el dispositivo está emitiendo señal.

No se ha incluido el pin de datos del módulo de frecuencia debido a que la librería trabaja automáticamente sobre el pin 12 si no se establece otro deliberadamente.

Métodos adicionales

enviar( )

En este método tiene lugar el envío del mensaje por radiofrecuencia (ilustración 50).

```
void enviar (char *mensaje)
{
  Serial.println("enviando mensaje");

  //envio del mensaje
  vw_send((byte *)mensaje, strlen(mensaje));
  //se espera hasta su completa transmisión
  vw_wait_tx();
  //muestra el mensaje por pantalla
  Serial.println(mensaje);
  Serial.println("Mensaje enviado\n");
}
```

Ilustración 50. Método de envío de mensajes

Simplemente se hace una llamada al método vw_send( ), perteneciente a la librería VirtualWire, que envía el mensaje establecido por parámetro, y espera a que se termine de transmitir.

Setup()

Se establece la comunicación por serial a 9600 baudios por segundo, la velocidad de transmisión a 5000 bits por segundo, y se muestra un mensaje por pantalla de iniciación del programa. Para finalizar, se enciende el LED que muestra que está funcionando. El método se muestra en la imagen 51.

```
void setup(){
  Serial.begin(9600);
  vw_setup(5000);
  Serial.println("Sistema iniciado\n");
  //retroalimentacion
  digitalWrite(pinLED, HIGH);
}
```

Ilustración 51. Método Setup del emisor

Loop()

Se llama a la función de enviar mensaje infinitas veces (ilustración 52).

```
void loop () {

  enviar("M3nS4J3");

}
```

Ilustración 52. Método Loop en el cual se establece el envío del mensaje

4.2 Implantación del software del vehículo

Librerías

Las librerías necesarias para el funcionamiento del sketch serán VirtualWire para la comunicación por radiofrecuencia, y NewPing para el uso del sensor ultrasónico. La inclusión de las librerías se muestra en la imagen 53.

```
#include <VirtualWire.h>
#include <NewPing.h>
```

Ilustración 53. Librerías controlador del vehículo

Métodos adicionales

int comp_colision( ):

Esta tarea se ocupará del sistema anti-colisión del vehículo.

Para ello, se hace una medición de la distancia detectada. Si lo detectado supera la distancia máxima permitida, se continuará con la siguiente tarea; pero si detecta un objeto a una distancia menor que la permitida, se iniciará el modo anti-colisiones.

Dicho modo funciona de la siguiente manera:

1. Se establece el valor de velocidad de los motores al máximo (255).
2. Se crea un valor aleatorio, el cual tendrá como valor 2 o 3.
3. Se gira durante un segundo hacia dicha dirección.
4. Se actualiza la distancia detectada. Si no hay obstáculos, se avanza un segundo hacia delante.
5. Se actualiza el valor de medición, volviendo a ejecutar esta misma secuencia si sigue habiendo obstáculos. Si no hay obstáculos se procede a la tarea de seguimiento.

De esta manera, se asegura que el vehículo no entre en bucle permaneciendo siempre en los mismos movimientos sin poder encontrar una salida.

int recibir_mensajes( )

Este método permitirá comprobar a cada receptor si recibe algún mensaje perteneciente al emisor. En total, cada uno podrá comprobarlo 10 veces como máximo. Esto se logra mediante el uso de un bucle "for", el cual se repetirá 40 veces repartiendo los turnos entre los 4 receptores.

Para analizar qué pautas de implementación se necesitan, se tiene que contar con los siguientes inconvenientes:

1. No es posible leer mensajes a la vez en todos los receptores con sólo una placa.
2. No se puede obtener directamente la intensidad de conexión entre los extremos, es decir, el emisor puede hacer llegar señal a dos o más receptores, pero no se sabrá cual está más cerca con los medios de los que se disponen.

Para el primer caso, la solución fue cambiar sucesivamente de receptor, de manera que, cada uno, únicamente pueda recibir 1 mensaje como máximo (0 si no le llega la señal), y así hasta que cada receptor haga 10 iteraciones.

No se ha tomado la opción de esperar los 10 mensajes seguidos en el mismo receptor, debido a que se puede dar el caso de que la señal no se transmita correctamente en un cierto intervalo de tiempo, penalizando de esta manera completamente al receptor actual respecto a los demás.

```
for(int  i=0; i<40;i++){
      if(sensor_actual==1){
        vw_set_rx_pin(recep_norte);

      }else if(sensor_actual==2){
        vw_set_rx_pin(recep_este);

      }else if(sensor_actual==3){
        vw_set_rx_pin(recep_oeste);

      }else if(sensor_actual==4){
        vw_set_rx_pin(recep_sur);

      }
```

Ilustración 54. Recepción de cada mensaje en cada receptor uno a uno

Como se puede observar en la imagen 54, en cada iteración del bucle se comprueba qué sensor es el actual mediante la variable “sensor_actual”, la cual se incrementa al final del bucle para, en la siguiente iteración, continuar con el siguiente receptor.

Para el segundo caso, referente a la intensidad de la conexión entre receptor-emisor, se crearon las variables que funcionan como contador para cada uno de los receptores. El proceso completo es el siguiente:

Si en el turno de un receptor en concreto, recibe un mensaje antes de que termine el tiempo de espera, aumentará su contador propio en una unidad. Al finalizar el bucle se tendrá el total de mensajes que ha recibido cada uno.

```
if (vw_get_message(mensaje, &mensajeLength)){
      digitalWrite(LEDDetecc, HIGH);
      if(comprobar_mensaje("M3nS4J3") == 0){
        switch(sensor_actual){
          case 1:
              cont_norte++;
            break;
          case 2:
              cont_este++;
            break;
          case 3:
              cont_oeste++;
            break;
          case 4:
              cont_sur++;
            break;
```

Ilustración 55. Contador de mensajes para cada receptor

Como se muestra en la imagen 55, cada vez que llegue un mensaje, se comprobará en primer lugar si pertenece al emisor auténtico (se da por hecho que ningún otro dispositivo emite el mismo mensaje), y en caso afirmativo se procederá a aumentar el contador correspondiente.

Para proveer de retroalimentación sobre la detección de señal por parte del vehículo, en el primer caso de que se reciba un mensaje, se encenderá el LED asociado, como se muestra en la imagen 56:

```
if (vw_get_message(mensaje, &mensajeLength)){
      digitalWrite(LEDDetecc, HIGH);
```

Ilustración 56. Activación retroalimentacón mensajes

int comparar( )

Esté método supone una ayuda para la comunicación, de manera que proporciona la función para saber si se trata del mensaje del emisor válido, dando por supuesto que nadie más conoce el mensaje que se transmite para verificar su autenticidad (ilustración 57).

```
int comprobar_mensaje(char* cadena) {
  //compara la cadena pasada por parametro con el mensaje recibido.
  //retorna 1 si son iguales y 0 si no

  for(int i = 0; i<mensajeLength; i++)
  {
    if(mensaje[i] != cadena[i])
    {
      return 1;
    }
  }

    return 0;
}
```

Ilustración 57. Comprobación autenticidad del emisor

int calcular_direccion( )

En este método tendrá lugar el cálculo de la dirección que tomará el vehículo en cada iteración de la tarea de seguimiento. Los valores de las distintas direcciones posibles serán las siguientes que se indican en la tabla 82, y el código empleado tiene lugar en la imagen 58:

Receptor	Valor dirección
Receptor norte	1
Receptor este	2
Receptor oeste	3
Receptor sur	4
Sin dirección	5

Tabla 82. Valor de cada dirección en el programa

Una vez se tiene el número de mensajes que ha llegado a cada receptor gracias al método recibir_mensajes( ), se obtiene el porcentaje de cada uno respecto al total. Para ello, se suman los cuatro contadores, y se almacena en una variable de tipo "float" el resultado de dividir cada contador con el total.

```
float total= cont_norte + cont_este + cont_oeste+ cont_sur;

if(total==0){
  parar();
  Serial.println("NO han llegado mensajes");
  return 5;
}else{

float por_norte=cont_norte/total;
float por_este=cont_este/total;
float por_oeste=cont_oeste/total;
float por_sur=cont_sur/total;
```

Ilustración 58. Obtención de los porcentajes de comunicación

El motivo del uso de porcentajes, es debido a que se necesita saber el grado de comunicación que tuvo cada uno respecto al total para poder seleccionar la dirección más óptima.

Antes del cálculo de los porcentajes, se comprueba si no ha llegado ningún mensaje: en caso afirmativo se devolverá el valor 5 perteneciente a la dirección nula para que el vehículo no haga ningún movimiento. Si el total de mensajes es mayor que 0, entonces se continúan con las siguientes instrucciones del cálculo de dirección.

Ahora que se tienen ya los porcentajes, se almacenan en un array. Dicho array será bidimensional, ya que se necesita tener relacionado el porcentaje con la dirección que está asociada a su receptor. La primera fila del array corresponderá a los porcentajes, mientras que la segunda fila será la dirección que le corresponde a cada receptor.

```
int pors[2][4]= {{por_norte,por_este,por_oeste,por_sur},{1,2,3,4}};
```

Ilustración 59. Array que mantiene la relación entre los porcentajes y su receptor correspondiente

A continuación se procede a ordenar el array por el método de la burbuja, ya que se necesita reordenar los porcentajes de mayor a menor para poder seleccionar de una manera más fácil la dirección a tomar, siendo la primera posición la del receptor que más porcentaje de comunicación tuvo. A la vez que se van ordenando los porcentajes, se van ordenando también las direcciones asociadas para no perder la integridad.

Finalmente ya sólo queda, en función de los valores del array, elegir la dirección óptima. Habrá dos casos distintos (ilustración 60):

1. Al haber 4 receptores, se considera que si el porcentaje de cada uno es menor que el 30% de la comunicación completa, significa que todos los receptores han tenido la misma comunicación, y por tanto se devolverá la dirección nula haciendo que el vehículo no haga ningún movimiento al estar al lado del emisor.

   Se ha elegido como valor el 30% debido a que, aunque la cuarta parte suponga un 25%, es el valor que provee un buen margen de comparación.
2. Si no se ha dado el caso anterior, directamente se coge la primera dirección del array, considerada la dirección que más se detectó durante el proceso de recibimiento de mensajes.

```
if(pors[0][0]< 0.3 && pors[0][1] <0.3 && pors[0][2]< 0.3 && pors[0][3]< 0.3){
   Serial.println("Cerca del emisor");
   return 5;
}else{
 Serial.println("Direccion tomada:");
 Serial.println(pors[1][0]);
 toleranciaFallos[contFallos]=pors[1][0];
 contFallos++;
 return pors[1][0];
}
```

Ilustración 60. Lógica de selección de la dirección a tomar

Parámetros globales

Pines de conexión para los motores (ilustración 61):

```
int IN1 = 13;
int IN2 = 12;
int IN3 = 11;
int IN4 = 10;
int ENA = 38;
int ENB = 39;
```

Ilustración 61. Pines de conexión de motores

Pines para los LEDs de retroalimentación (ilustración 62):

```
int LEDDetecc = 22;
int LEDColision = 24;
```

Ilustración 62. Pines para la retroalimentación

Parámetros para el sistema anti-colisiones:

Para el empleo del componente ultrasónico, serán necesarios por una parte los pines de conexión a la placa Arduino ya establecidos anteriormente en el apartado de diseño. Su definición se muestra en la imagen 63

```
#define TRIGGER_PIN  2
#define ECHO_PIN     3
```

Ilustración 63. Conexión de pines del sensor ultrasónico

Además, debido a que se ha seleccionado la librería NewPing para su uso, serán necesarios los siguientes parámetros para poder medir las distancias (ilustración 64).

```
#define MAX_DISTANCE 100
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

Ilustración 64. Parámetros necesarios para la medir la distancia

El parámetro llamado “MAX_DISTANCE” establece la distancia máxima que puede detectar. Se ha fijado en 100 ya que 100 centímetros es una distancia más que suficiente para que el vehículo no colisione, teniendo en cuenta la velocidad máxima a la que se puede desplazar.

Por último, se establece la distancia máxima a la que se podrá acercar un objeto detectado. En la futura implementación, este valor será de 40 centímetros (ilustración 65).

```
int max_dist_colision = 40;
```

Ilustración 65. Distancia máxima acercamiento a objeto

Parámetros para la comunicación por radiofrecuencia:

En primer lugar, se establecen los pines para los receptores. Además, se crea un contador para cada uno, en el cual se contarán el número de mensajes que ha recibido cada uno de ellos cada vez que se ejecute la tarea. Esta definición de parámetros tiene lugar en la imagen 66

```
int recep_norte = 9;
int recep_este = 8;
int recep_oeste = 7;
int recep_sur = 6;

float cont_norte=0;
float cont_este=0;
float cont_oeste=0;
float cont_sur=0;
```

Ilustración 66. Pin de cada receptor y su propio contador

Se ha creado también una variable (ilustración 67) que servirá para controlar el sensor que está recibiendo los mensajes en el momento, y otra para establecer el tiempo máximo que podrá estar esperando un receptor para recibir un mensaje. Si pasado dicho tiempo, no ha recibido ninguno, se continuará con la ejecución.

Para terminar con las variables pertenecientes a la comunicación, se finaliza declarando las que contendrán el contenido del mensaje, tanto la cadena de dígitos como su longitud. La longitud se ha fijado en 7 debido a que la cadena “M3nS4J3” contiene ese número de caracteres (ilustración 67).

```
int sensor_actual=1;
const int timeout_sms = 60;

byte message[7];
byte messageLength = 7;
```

Ilustración 67. Parámetros extras para la comunicación

Se necesitará un parámetro que controle el tiempo que durará cada desplazamiento durante la tarea de seguimiento. Se diferenciarán dos tiempos, uno para el desplazamiento durante la tarea de seguimiento, y otro para el sistema de colisiones. El tiempo para el sistema anti-colisión será menor debido a la existencia comprobada de obstáculos alrededor. Esta definición de tiempos tiene lugar en la imagen 68. Para la tarea de desplazamiento, dispondrá de 1,5 segundos, ya que se trata de un estado en el que todavía no se han detectado obstáculos, y se puede dar mayor margen a la maniobra.

Por el contrario, para la tarea de sorteo de obstáculos, se ha dado 1 segundo para el movimiento, ya que al estar en dicho estado, se requiere una mayor frecuencia de actualización en referencia a los obstáculos del entorno, y así poder empezar cuanto antes la tarea de seguimiento de nuevo.

```
int tiempo_despl_seguimiento = 1500;
int tiempo_despl_colision = 1000;
```

Ilustración 68. Tiempo según tipo de desplazamiento

Por último, se muestran los parámetros para el manejo de la tolerancia a fallos (ilustración 69). Consta de un array de dos posiciones, donde se irán actualizando las dos últimas direcciones tomadas. También se incluye un contador que se utilizará para saber en qué posición del array se encuentra la dirección última calculada, ya que podrá ser cualquiera de las dos posiciones existentes.

```
int toleranciaFallos[2]= {5,5}; //5 es la direccion nula
int contFallos=0;
```

Ilustración 69. Parámetros tolerancia a fallos

int comprobarFallo( )

Este método ha sido necesario debido a la existencia del hándicap que provoca que la comunicación por radiofrecuencia entre este tipo de módulos no esté asegurada. Esto

quiere decir que, si por ejemplo, se tiene un escenario donde el emisor está situado a la misma distancia del receptor norte y del receptor oeste, puede darse el caso de que al receptor norte le lleguen todos los mensajes posibles en ese lapso de tiempo, pero no llegue ninguno al otro. Por este motivo, la recepción de mensajes y su posterior gestión está basada en probabilidades.

Para solucionar este problema, se ha diseñado un modo a prueba de fallos, el cual consiste en lo siguiente:

En primer lugar, actualmente se tiene un array de int 2 posiciones como parámetro global, en las cuales se almacenan las 2 últimas direcciones calculadas, incluyendo entre estas la de la iteración actual.

Los casos por los que se considerará que se trata de una dirección errónea serán cuando: la dirección anterior sea adelante y la nueva hacia atrás y viceversa, aplicándose la misma política para los casos derecha-izquierda.

En caso de que se dé alguno de los casos mencionados, se volverá a tomar la dirección anterior, y sólo si vuelve a darse la misma dirección se acabará tomando como válida.

En el resto de casos, se considerará correcta la dirección calculada.

int mover_motores( )

Para esta funcionalidad, ya se conocen los pines de la placa a los que irán conectados cada una de las conexiones del controlador de motores que se encargan de moverlos (IN1, IN2, IN3, IN4).

Este método, simplemente hace las llamadas a los métodos que establecen los valores en los pines de cada motor, de manera que ejercen el movimiento seleccionado.

Una vez se acciona el movimiento, se espera el tiempo definido por parámetro, y se detienen los motores.

Setup()

Se establecen los pines del motor y la comunicación con el ordenador para mostrar la información por pantalla y se inicia el programa (ilustración 70).

```
void setup()
{
  pinMode (IN1, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (IN4, OUTPUT);

  pinMode (ENA, OUTPUT);
  pinMode (ENB, OUTPUT);

  Serial.begin(9600);
  Serial.println("Iniciando...");

  vw_setup(5000);
  vw_rx_start();
}
```

Ilustración 70. Método Setup controlador del vehículo

Loop()

Como se explicó en el apartado 3.2.2, en el loop deberá ir la secuencia establecida. Se comenzará llamando a la tarea de comprobación de colisiones. Cuando termine, se hará una lectura de mensajes de forma que cada receptor lo haga 10 veces como máximo. Para ello, se llama en total 10 veces al método destinado a recibir mensajes. A continuación, se calcula la dirección y se resetean los contadores de los mensajes para la siguiente iteración, se comprueba la dirección calculada, y finalmente se accionan los motores hacia dicha dirección.

Se ha escogido hacer 10 iteraciones debido a que es el valor a partir del cual los resultados presentaban mayor fiabilidad estadísticamente. La definición del método se muestra en la imagen 71.

```
void loop()
{


    comprobar_colision();

    for(int i=0;i<10;i++){
          recibir_mensajes();

    }
    digitalWrite(LEDDetecc, LOW);

    direccion_actual = calcular_direccion();
    //reseteamos contadores para siguiente iteracion
    cont_norte=0;
    cont_este=0;
    cont_oeste=0;

    if(direccion_actual != 5){
      direcc_comprobada = comprobarFallo(direccion_actual);

      analogWrite(ENA,128);
      analogWrite(ENB,128);
      mover_motores(direcc_comprobada, tiempo_despl);
    }
  }
```

Ilustración 71. Método loop controlador del vehículo

4.3 Pruebas

Una vez se ha terminado con toda la implementación, es el momento de hacer las pruebas pertinentes para asegurar que todo funciona como es debido.

Cada prueba seguirá el siguiente formato mostrado en la tabla 83:

Identificador: PF-xx (nombre)	
Sistema	**Software/Hardware**
Resultado	
Descripción	
Acciones:	

Tabla 83. Formato de prueba funcional

Donde cada campo representa:

- Identificador: nombre único y breve de la prueba funcional en cuestión. 'xx' representa el número de prueba, como por ejemplo, PF-02.
- Sistema: elemento sobre el que se hace la prueba. Podrá ser sobre el Software o sobre el Hardware.
- Descripción: descripción completa de la prueba en cuestión expresada lo más claramente posible.
- Acciones: pasos que tiene lugar en cada prueba.

Identificador: PF-01 (Los motores ejercen movimiento)	
Sistema	Hardware
Resultado	Correcto
Descripción	Comprobación de que los motores son capaces de desplazar al vehículo
Acciones:	1. Se provee de corriente a los motores 2. Se establecen sus pines de todas las maneras posibles para verificar que se giran en ambos sentidos

Tabla 84. Prueba funcional PF-01

Identificador: PF-02 (Feedback LEDs)	
Sistema	Hardware
Resultado	Correcto
Descripción	Verificación de que cada LED se enciende y se apaga correctamente en la tarea precisa.
Acciones:	1. Comprobación de que el LED del emisor se enciende al activar el dispositivo. 2. Comprobación de que el LED del sistema anti-colisiones se enciende cuando detecta un elemento dentro del límite. 3. Comprobación de que el LED del sistema de detección se enciende en cuanto recibe un mensaje algún receptor.

Tabla 85. Prueba funcional PF-02

Identificador: PF-03 (Receptores reciben señal)	
Sistema	Hardware
Resultado	Correcto
Descripción	Comprobación de que los receptores reciben señal proveniente del emisor
Acciones:	1. Encendido del dispositivo emisor 2. Verificación de que a los receptores les llegan mensajes.

Tabla 86. Prueba funcional PF-03

Identificador: PF-04 (Vehículo se desplaza hacia el emisor)	
Sistema	Software
Resultado	Correcto
Descripción	Mediante toda la implementación hecha, se comprueba que el vehículo finalmente detecta al emisor, y avanza en su dirección. Debido a que la estimación de la dirección se rige por probabilidades, se busca tener la mayor precisión posible para su cálculo
Acciones:	1. Se envía señal desde el emisor 2. Se comprueba que los receptores reciben los mensajes y se calcula la dirección, de manera que se desplaza hacia su localización

Tabla 87. Prueba funcional PF-04

Mediante continuas pruebas se ha estimado la tasa de aciertos que se produce de media en la ilustración 72. El resultado es satisfactorio teniendo en cuenta el hecho de trabajar con probabilidades y componentes de bajo coste.

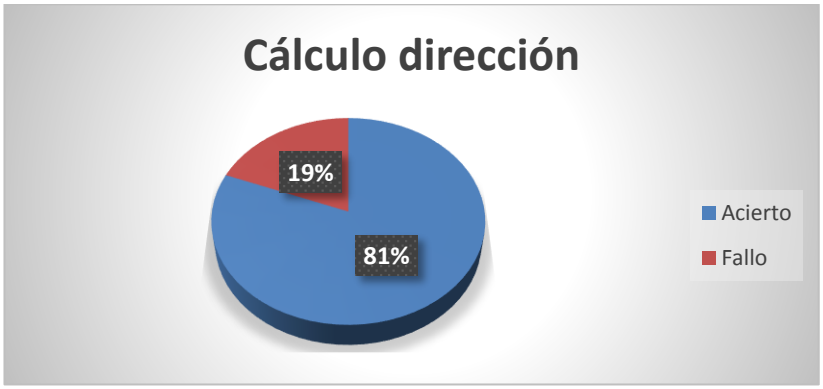

Ilustración 72. Porcentaje de aciertos en cálculo de dirección

Identificador: PF-05 (Sistema anti-colisiones)	
Sistema	Software/Hardware
Resultado	Correcto
Descripción	Verificación de que el vehículo esquive los obstáculos que se encuentre
Acciones:	1. El vehículo se encuentra de frente con un obstáculo 2. Por medio del sistema anti-colisiones, el vehículo sortea el obstáculo tomando otra dirección que no sea hacia él, hasta evitarlo finalmente

Tabla 88. Prueba funcional PF-05

Identificador: PF-06 (Vehículo inmóvil al lado del emisor)	
Sistema	Software
Resultado	Correcto
Descripción	Verificación de que el vehículo no se desplaza hacia ninguna dirección al estar cerca del emisor
Acciones:	1. Se espera a que el vehículo se aproxime al emisor o nos aproximamos hacia él 2. Comprobamos que, aunque esté activo el seguimiento del vehículo, al estar cerca permanece inmóvil

Tabla 89. Prueba funcional PF-06

Identificador: PF-07 (Dirección errónea)

Sistema	Software
Resultado	Correcto
Descripción	Al detectar una nueva dirección totalmente contraria a la anterior, se vuelve a ejecutar la anterior ya que se tomará como no válida en principio. Si se repite, ya se tomará por correcta
Acciones:	1. Se calcula una dirección contraria a la de la iteración anterior 2. Se ignora y se toma la dirección anterior 3. Si se repite la misma, ya se toma como válida y se elige como la dirección para el desplazamiento

Tabla 90. Prueba funcional PF-07

Identificador: PF-08 (Mensajes no auténticos)	
Sistema	Software/Hardware
Resultado	Correcto
Descripción	Comprobación de que si no se envían los mensajes con la cadena establecida, se ignoran para el cálculo de la dirección.
Acciones:	1. Se envían mensajes sin la cadena establecida 2. Se comprueba que no surten efecto en el controlador.

Tabla 91. Prueba funcional PF-08

4.4 Matriz de trazabilidad de pruebas

	RS01	RS02	RS03	RS04	RS05	RS06	RS07	RS08	RS09	RS10	RS11	RS12	RS13	RS14	RS15	RS16	RS17	RS18	RS19	RS20	RS21	RS22
PF01	x				x					x			x	x	x	x	x	x				
PF02						x	x	x			x	x		x								
PF03				x							x	x		x	x	x			x	x	x	
PF04		x			x						x		x	x	x	x			x	x	x	x
PF05		x			x				x				x	x								
PF06		x	x		x						x	x		x								
PF07														x								x
PF08														x								x

Tabla 92. Matriz de trazabilidad de pruebas con requisitos funcionales

Capítulo 5. Planificación y presupuesto

En este capítulo, primero tendrá lugar toda la tarea de planificación que se ha seguido durante la realización del proyecto. Se mostrará un desglose de cada tarea con su duración, además de un diagrama de Gantt para proporcionar una descripción más visual.

Para finalizar el capítulo, se detallará el presupuesto que ha sido necesario para su realización, separándolo por categorías para, finalmente, mostrar el presupuesto global.

5.1 Planificación

Para hacer la planificación, se ha tenido un seguimiento continuo del trabajo hecho, para dejar constancia del tiempo dedicado de la manera más exacta posible.

5.1.1 Planificación de tareas

Actividad	Fecha inicio	Fecha fin	Duración (días)
Propuesta	14/06/16	15/06/16	2
Presupuesto	16/06/16	26/06/16	11
Estudio de viabilidad	27/06/2016	10/07/2016	15
Análisis	11/07/2016	26/07/2016	16
Diseño	29/07/2016	23/08/2016	27
Implementación	24/08/2016	08/10/2016	47
Pruebas	09/10/2016	20/10/2016	12
Documentación	24/10/2016	15/12/2016	53

Tabla 93. Planificación de tareas

5.1.2 Diagrama de Gantt

A continuación, en la ilustración 73 se muestra el diagrama de Gantt sobre el desarrollo de todo el proyecto dividido en tareas.

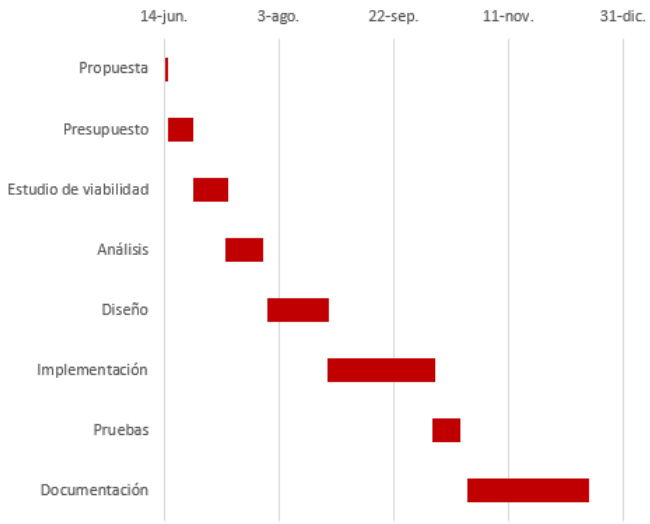

Ilustración 73. Diagrama de Gantt

5.2 Presupuesto

En este segundo apartado del capítulo, se va a llevar a cabo una aproximación del presupuesto que ha sido necesario para este proyecto. Se organizará por diferentes categorías, siendo éstas costes de personal, costes de software y costes de hardware, para proporcionar finalmente una estimación de los gastos totales.

Cada precio indicado tiene el IVA incluido.

5.2.1 Resumen de horas dedicadas

En primer lugar, se presenta en la tabla 94 el desglose de las horas dedicadas a cada una de las tareas que comprenden el proyecto.

Actividad	Días	Horas/día	Total
Propuesta	2	6	12
Presupuesto	11	4	44
Estudio de viabilidad	15	4	60
Análisis	16	5	80
Diseño	27	4	108
Implementación	47	4	188
Pruebas	12	3	36
Documentación	53	3	159
Total			687

Tabla 94. Resumen de horas dedicadas

5.2.2 Costes de personal

En segundo lugar, en la tabla 95 se muestra una estimación del coste total debido a las horas dedicadas a la realización del proyecto.

Concepto	Apellidos, nombre	Horas dedicadas	Coste por hora	Total
Analista	Carrera Martín, Alberto	391	30€	11730€
Diseñador	Carrera Martín, Alberto	108	24€	2592€
Programador	Carrera Martín, Alberto	188	20€	3760€
Total				18082€

Tabla 95. Costes de personal

5.2.3 Costes de software

En la tabla 96 se recoge el coste individual y total de los recursos software utilizados.

Concepto	Cantidad	Precio unitario	Total
Windows 7 Ultimate 64 bits	1	140€	140€
Microsoft Office Proffesional Plus 2013	1	190€	190€
Total			330€

Tabla 96. Costes de software

No se ha incluido el resto de software utilizado debido a que son de uso gratuito. Dicho software excluido se menciona a continuación:

- Arduino IDE
- Fritzing

5.2.4 Costes de hardware

En este caso, la tabla 97 muestra el coste de los recursos hardware.

Concepto	Cantidad	Precio unitario	Amortizado	Cantidad	Total
Ordenador de sobremesa	1	890€	118,11€	1	118,11€
Pantalla	1	150€	15€	1	15€
Placa Arduino Mega 2560	2	10,20€	-	2	20,40€
Protoboard	2	1,93€	-	2	3,86€
Chasis vehículo a escala con ruedas y motores	1	12,41€	-	1	12,41€
65x cables macho-macho para Arduino	1	2,30€	-	1	2,30€
40x cables macho-hembra para Arduino	1	3,99€	-	1	3,99€
5x kit RF	1	6,99€	-	1	6,99€
Módulo controlador de motores L298N	1	3,26€	-	1	3,26€
Sensor ultrasónico HC-SR04	1	1,80€	-	1	1,80€
Portapilas	2	2.50€	-	2	5€
Total					193,12€

Tabla 97. Costes de hardware

5.2.5 Costes totales

Finalmente, se puede observar en la tabla 98 los costes totales que han tenido lugar para elaborar el proyecto completamente.

Concepto	Total
Costes de personal	18082€
Costes de Software	330€
Costes de Hardware	193,12€
Total	18605,12€

Tabla 98. Costes totales

Capítulo 6. Conclusiones y líneas futuras

6.1 Conclusiones generales

Como conclusiones globales sobre el proyecto se puede destacar, principalmente, el hecho de haber conseguido la autonomía de un vehículo para triangular por medio de radiofrecuencia, algo que se no se consideraba desde un primer momento como posible al 100%.

Esto ha sido posible gracias al previo estudio y comprensión del funcionamiento de los componentes de radiofrecuencia que, ayudado del diseño posterior de su comunicación, hizo factible dicho objetivo.

Además de este propósito cumplido, se han logrado otros tantos que, a pesar de saberse desde el comienzo que eran posibles de realizar, cada uno de ellos tenía cierta complejidad a la hora de su desarrollo. Estas metas fueron las siguientes:

- ✓ Montaje del chasis que sirvió de estructura para el vehículo, teniendo que realizarse a partir de 0, contando únicamente con las piezas para su construcción.
- ✓ Desplazamiento del vehículo gracias a los motores, conseguido mediante un suministro correcto de energía a los mismos.
- ✓ Desarrollo de los controladores Arduino, consiguiendo así controlar todo el funcionamiento tanto del vehículo como del dispositivo emisor.

Globalmente a todos ellos, también se ha conseguido llevar un adecuado seguimiento de cada tarea realizada perteneciente a todo el proyecto, de manera que la ejecución de cada una de éstas se cumplía dentro del tiempo establecido y de la forma previamente pensada para considerar que, efectivamente, la tarea había sido realizada con éxito.

A lo largo del proyecto, también tuvieron lugar contratiempos en cuanto a problemas que surgieron, tanto con los componentes hardware como en el software, habiéndose solucionado todos correctamente. Un ejemplo de ellos fue el imprevisto de que llegaran defectuosas las piezas encargadas de la comunicación por radiofrecuencia, lo cual frenó la implantación del sistema y la recogida de pruebas.

Finalmente, aunque se consiguió el objetivo principal de la autonomía del vehículo para la tarea de seguimiento, al ser los componentes de radiofrecuencia tan delicados, y

con diferencias de fiabilidad entre todos ellos, provocaron que el resultado final en el vehículo no fuese el comportamiento ideal al 100%, ya que había unos receptores que, por lo general, no terminaban de recibir los mismos mensajes que los demás, aunque estuviesen en las mismas condiciones de distancia respecto a los otros. Aun así, con unos componentes altamente fiables los resultados arrojarían un comportamiento igual y si no casi idéntico al que se esperaba.

6.2 Conclusiones personales

Personalmente, creo que lo más complicado de lograr en cualquier proyecto que nos planteamos es la constancia y el trabajo diario. Con metas así, logramos mejorar la confianza en lo que hacemos y en nuestras propias expectativas y capacidades y, además, considero que este proyecto me ha servido para ir más allá de cualquier otra práctica que he realizado a lo largo de la carrera.

Por primera vez, dependía de mí mismo para hacer un trabajo de tal importancia en su ejecución, para el cual tenía que organizarme, y llevar a cabo un proceso autónomo (excluyendo la ayuda recibida por mi tutor) y así conseguir cada una de las partes que comprenden todo el proyecto, tales como investigar componentes, establecer los requerimientos, diseñar e implementar todo el sistema, etc.

Obviamente, también sería necesario mencionar el proceso de aprendizaje que seguimos con trabajos de esta envergadura; no es lo mismo que nos proporcionen información acerca de un tema y tengamos que estudiarlo, que si esa información tiene que ser buscada y elaborada por nosotros mismos. Desde luego, al adquirir conocimientos de esta forma, nos empapamos más acerca del tema que hemos seleccionado para trabajar sobre él, porque al fin y al cabo en el proyecto sólo queda reflejada una pequeña parte de esa investigación que hemos llevado a cabo para conseguir toda la información que se ha considerado necesaria y útil.

En definitiva, un trabajo de este estilo te ayuda a superarte a ti mismo, a mejorar en todos los aspectos en referencia a la autonomía, y sobre todo a hacerte crecer como persona para llevar a cabo tareas que suponen un gran coste de esfuerzo y constancia.

6.3 Líneas futuras

6.3.1 Motor paso a paso

El motor paso a paso consiste en un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, por lo tanto presenta una serie de ventajas como por ejemplo lograr la repetitividad en cuanto al posicionamiento.

Sobre este tópico, se pueden encontrar entre sus aplicaciones más comunes la robótica o los drones, o en este caso su uso en radiocontrol que ha sido la base de este proyecto.

Reflexionando sobre el tema, sería posible cambiar el tipo de motores actuales (que siguen una corriente continua) por motores paso a paso, para lograr una mejor precisión sobre el posicionamiento.

La mayor diferencia entre unos y otros radica en que los primeros convierten la energía eléctrica y mecánica consiguiendo así un movimiento rotatorio gracias a la acción que se genera en el campo magnético; pero resulta difícil conseguir una óptima variación de la velocidad de rotación, ya que dependen de la tensión de suministro. Además los pulsos electromagnéticos que genera pueden afectar a ciertos equipos electrónicos sensibles como por ejemplo las radios o cualquier sistema que opere con radiofrecuencias (dado que éstas se propagan en el aire), y otro inconveniente además es que el mantenimiento de este tipo de motores es caro.

Por el contrario, los motores paso a paso usan microprocesadores que permiten controlar con gran precisión el giro ejercido por las ruedas, y por tanto logran un mayor control de todos los desplazamientos que tienen lugar sobre el vehículo. Además el ángulo de rotación que se consigue es proporcional a los pulsos de entrada que recibe, por lo que se puede conseguir un gran rango de velocidades de rotación. Por último entre otras ventajas, presenta una buena respuesta de arranque y parada.

6.3.2 Vehículo seguidor de línea

También se podría establecer en el vehículo un componente sensor de infrarrojos, el cual detectase una línea dibujada en el suelo y, mediante la implementación correspondiente, hiciese que el vehículo la siguiese por sí mismo. Estos robots poseen sensores para detectar dichas líneas dibujadas, usando motores de paso a paso por

ejemplo, que funcionan gracias a la energía proporcionada por baterías o corriente alterna.

6.3.3 Control manual del vehículo por radiofrecuencia

Además de usar la radiofrecuencia en este caso de manera autónoma en el vehículo, también se podría utilizar para enviar señales al vehículo con otras acciones diferentes, tales como hacer que se desplace hacia adelante, gire, etc.

6.3.4 Control del vehículo vía wifi

Mediante un componente Ethernet instalado en el vehículo, se podría conseguir que, mediante una interfaz web, se pudiesen mandar acciones a ejecutar sobre el vehículo para conseguir su control remoto por un medio distinto a la radiofrecuencia.

Apéndice 1: Acrónimos

La tabla 99 ayudará a comprender los acrónimos que se encuentran a lo largo del documento.

Acrónimo	Definición
ASK	Amplitude-Shift Keying
DC	Corriente continua
EEPROM	Electrically Erasable Programmable Read-Only Memory
E/S	Entrada/Salida
Gps	Global Positioning System
DC	Direct Current
LED	Light-Emiting Diode
Ma	Miliamperio
Mhz	Megahercio
PWM	Pulse-Width Modulation
KB	Kilobyte
RAM	Random Access Memory
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TWI	Two Wired serial Interface
USB	Universal Serial Bus
V	Voltio

Tabla 99. Acrónimos

Apéndice 2: Definiciones

Este apartado tiene como objetivo servir como diccionario para aquellos términos que se desconozcan. Dichos términos son recogidos en la tabla 100.

Término	Definición
Bootloader	Programa que prepara lo necesario para iniciar un sistema
Bus	Canal que transfiere datos entre componentes
Hardware	Partes físicas tangibles de un sistema informático
Hardware libre	Dispositivo hardware cuyas especificaciones y diagramas esquemáticos son de acceso público
Java	En este documento se menciona como lenguaje de programación
Jumper	Elemento que permite cerrar el circuito eléctrico del que forma parte
Memoria flash	Memoria que permite la lectura y escritura de múltiples posiciones de memoria en la misma operación
Microcontrolador	Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria
Pin	Patilla de cada uno de los contactos metálicos de un componente fabricado de un material conductor de la electricidad
Radiofrecuencia	Cada una de las frecuencias de las ondas electromagnéticas empleadas en la radiocomunicación
Sketch	Nombre característico de los programas implementados en Arduino
Software	Parte lógica de un sistema informático

Tabla 100. Definiciones

Apéndice 3: Referencias

En este apéndice final se van a mencionar todas las referencias que han servido para ahondar en los conocimientos adquiridos a lo largo de todo el proceso de investigación y que aparecen en este documento.

Castro, A. Raspberry Pi. (2014). Recuperado de http://computerhoy.com/noticias/hardware/que-es-raspberry-pi-donde-comprarla-como-usarla-8614

Clasificación requisitos. Recuperado de http://www.ie.inf.uc3m.es/grupo/docencia/reglada/psi/unidad6-DOC.pdf

Controlador de motores L298N. Recuperado de http://www.naylampmechatronics.com/blog/11_tutorial-de-uso-del-modulo-l298n.html

Daniel. Documentación Kit RF. Recuperado de http://www.instructables.com/id/Modulos-RF433-Mhz-con-Arduino/

Detalles Placa Arduino Mega 2560. Recuperado de https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Detalles protoboard. (2008). Recuperado de http://www.circuitoselectronicos.org/2007/10/el-protoboard-tableta-de-experimentacin.html

Funcionamiento motor de corriente continua. Recuperado de http://www.prometec.net/motorcc/

Guía redacción casos de uso. Recuperado de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

Librería NewPing. Recuperado de http://playground.arduino.cc/Code/NewPing

Llamas, L. (2015). Guía uso sensor ultrasónico. Recuperado de http://www.luisllamas.es/2015/06/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

Maldonado, C. (2014). Novena. Recuperado de https://hipertextual.com/archivo/2014/08/novena-laptop-libre/

Martínez, F. (2015). Tutorial Arduino IDE. Recuperado de https://openwebinars.net/blog/tutorial-arduino-ide-arduino/

McCauley, M. Librería Virtual Wire. Recuperado de https://www.pjrc.com/teensy/td_libs_VirtualWire.html

Módulo controlador de motores Ardumoto. Recuperado de http://tienda.bricogeek.com/shields-arduino/429-ardumoto-driver-de-motores-para-arduino.html?gclid=CNyX1eO1i9ACFUa3GwodwXANog

Módulo inalámbrico RF APC220. (2015). Recuperado de http://www.tuelectronica.es/tutoriales/arduino/arduino-y-modulo-inalambrico-rfapc220.html

Noble, Joshua (15 de julio de 2009). Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFramework (1ª edición)

Ortiz, J. (2006). Funcionamiento kit RF (modulación ASK). Recuperado de http://www.mailxmail.com/curso-modulador-demodulador-digital-fsk-ask/modulacion-ask

Plaza, A. (2014). Ejemplos de plataformas de hardware libre. Recuperado de http://toyoutome.es/blog/10-frutos-del-hardware-libre/28730

Programación en Ardublock y MiniBloq. Recuperado de https://programamos.es/entornos-graficos-de-programacion-con-arduino/

Puente H. Recuperado de https://itp.nyu.edu/physcomp/labs/motors-and-transistors/dc-motor-control-using-an-h-bridge/

Reprap Project. (2016). Recuperado de https://all3dp.com/history-of-the-reprap-project/

Rosillo, P. (2014). Project Ara. Recuperado de https://andro4all.com/2014/04/informacion-project-ara-telefonos-modulares-revolucion-movil

Sensor ultrasónico HC SR04. Recuperado de http://electronilab.co/tienda/sensor-de-distancia-de-ultrasonido-hc-sr04/

Tutorial Fritzing. Recuperado de http://www.programacionyrobotica.com/fritzing/

Uso de motores y controlador de motores. Recuperado de http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/